Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Técnica en Informática de Gestión

Proyecto Final de Carrera

Implementación de un módulo de gestión de mapas utilizando herramientas de software libre

Autor: Alberto Anta Andrés

Tutor: Prof. D. Javier Ortiz Laguna

Fecha: Abril 2011

Título: Aplicación

Asunto: Memoria del Proyecto Fin de Carrera de Ingeniería Técnica en Informática de Gestión.

Autor: D. Alberto Anta Andrés

Tutor: Prof. D. Javier Ortiz Laguna

Universidad Carlos III de Madrid

Campus de Leganés

Agradecimientos

- A mis padres y mis hermanos, en especial a mi madre, por todo lo que ha luchado en la vida por sus hijos, sobre todo en los momentos de duros de soledad.
- A mis abuelos, Miguel y Catalina, por todo ese cariño, tiempo y ayuda que me han dedicado.
- A Laura, por estar siempre a mi lado en todo momento, por ser un apoyo constante, y a esa nueva vida que vamos a emprender juntos.
- A Raúl Rivero, por permitirme conocer el apasionante mundo de las redes y brindarme la oportunidad de poder formar parte su un equipo técnico.
- A Raúl, Óscar, Pepe, Nacho, Pablo y Héctor, por compartir conocimientos y sobre todo esos buenos ratos que pasamos todos los días.
- A Ángel y Javier, por esos días inolvidables en la Universidad.
- A toda la comunidad del Software Libre, en especial a su filosofía en el mundo de la computación.
- A Sésnandez de Tábara, ese remanso de paz de la sierra zamorana, mis orígenes, a sus gentes y su sencillez, nunca olvidaré esos veranos inolvidables.
- A Carranque, pueblo situado en la provincia de Toledo, a su gente por acogerme gentilmente y por ese futuro que representa en mi vida.
- A Freddie, Brian, Rodger y John, por tantos momentos disfrutando de sus arte y legado.

Índice de contenido

Índice de ilustraciones

1. INTRODUCCIÓN

Las soluciones para muchos problemas frecuentemente requieren acceso a varios tipos de información que sólo pueden ser relacionadas por geografía o distribución espacial. Sólo la tecnología de los "Sistemas de Información Geográfica" de aquí en adelante SIG, permite usando la geografía, almacenar y manipular información para analizar patrones, relaciones, y tendencias para contribuir a tomar mejores decisiones.

Cerca del 80% de la información tratada por instituciones y empresas publicas o privadas tienen en alguna medida relación con datos espaciales, lo que demuestra que la toma de decisiones depende en gran parte de la calidad, exactitud y actualidad de esta información espacial.

Los SIG se han constituido durante los últimos veinte años en una de las mas importantes herramientas de trabajo para investigadores, analistas y planificadores, en todas las actividades que tienen como objetivo el manejo de la información relacionada con diversos niveles de agregación espacial o territorial. Aunque los SIG tienen gran capacidad de análisis, éstos no pueden existir por si mismos, deben tener una organización, equipamiento y personal responsable para su implementación y mantenimiento.

Entre otras funcionalidades, los SIG deben poder proporcionar al usuario de las herramientas necesarias para poder acceder a la información en base a ciertos parámetros, como por ejemplo la georreferenciación o geolocalización1.

No obstante, el acto de geolocalizar ha ido más allá de las especialidades de geociencias y de SIG, en gran parte, debido a la aparición en los últimos años de nuevas herramientas, cuya facilidad de uso ha extendido y democratizado esta tarea fuera del ámbito técnico.

El uso de herramientas como Google Earth, o Yahoo Maps ha implicado un salto cualitativo en cuanto a geolocalización. Ya no se trata solamente de geodatos limitados a los especialistas de las geociencias y Sistemas de Información Geográfica. Ahora la geolocalización tiene un impacto sociológico puesto que se realiza sobre todos los contenidos sociales presentes en el mundo.

Un claro ejemplo de uso de un SIG fuera del ámbito estrictamente científico, es la implementación de servicios remotos que actualmente ofrecen distintos proveedores para cálculo de rutas o visualización de mapas y sin ir más lejos, la implementación de un servidor de geolocalizaciones y

mapas en el presente proyecto final de carrera, englobado dentro de un Sistema de Información Turística.

1.1 Descripción del sistema de información turística

El sistema en el que se englobará la implementación de un servidor de geolocalización y mapas, como subsistema o módulo del mismo, es un sistema que, tras su puesta en explotación, será capaz de ofrecer a los dispositivos móviles (teléfonos, PDA, PC, netbooks, tablets ...) que lo soliciten, un mapa urbano detallado de una serie de localizaciones correspondientes a monumentos y puntos de interés turísticos previa planificación de rutas específicas para cada dispositivo conectado.

Dada la cantidad de servicios y funcionalidades, el sistema se encuentra dividido en una serie de subsistemas o módulos interconectados entre sí, que ejecutan de forma eficiente y estructurada los procesos necesarios para obtener los resultados adecuados a las solicitudes de los dispositivos móviles interconectados.

Desde una vista general, los módulos que conforman el sistema son:

- Módulo o aplicación de los clientes (usuarios finales) residente en los dispositivos móviles. Aplicación con interfaz gráfica (en ingles GUI) que se encarga de solicitar los itinerarios y mapas de los puntos de interés turísticos a visitar por un usuario final.
- Módulo de comunicación. Implementado por una parte en el dispositivo móvil y en el servidor del sistema. Su misión es gestionar y enviar todas las comunicaciones entre los dispositivos móviles y los distintos módulos del servidor y poder devolver los resultados solicitados por los dispositivos móviles.
- Módulo de geolocalización y gestión de mapas. A partir de las peticiones de los clientes, obtiene la geolocalización y un mapa para enviarla a los dispositivos móviles.
- Módulo de gestión de monumentos. Se encarga del mantenimiento y tratamiento de todos los lugares de interés turístico del sistema de información.
- Módulo del planificador de rutas. Es el corazón del sistema, se encarga de procesar una lista de monumentos y calcular las rutas óptimas para su visita (desde horarios pasando por transportes y duración de los trayectos).
- Módulo de gestión de transportes. Se encarga de calcular los tiempos entre localizaciones

específicas y diferentes medios de transportes.

Estos módulos explicados, módulo de gestión de mapas, módulo de gestión de monumentos, módulo de gestión de transportes y planificador de rutas, se encontrarían desplegados en el servidor del sistema, al igual que el módulo de comunicación, ya que la aplicación de los usuarios finales formará parte de la implementación en los dispositivos móviles.

A continuación se detalla la totalidad del funcionamiento del sistema y todos su módulos.

1. El módulo de gestión de monumentos obtiene los monumentos desde internet o desde aplicaciones externas, tales como puntos de interés contenidos en aplicaciones de guiado GPS para coches.
2. Se insertan todos los datos de los puntos de interés turístico en una base de datos, La información contenida se corresponde con localización y dirección, horarios, precios, etc.
3. El dispositivo móvil solicita los monumentos disponibles. Los almacena en ficheros o memoria auxiliar. No es necesario hacerlo todos los días sino una o varias veces al mes, ya que, presumiblemente no habrá muchas modificaciones de monumentos en la base de datos.
4. El usuario final del dispositivo móvil, seleccionará los monumentos que quiere visitar, incluso asignando prioridades entre ellos. Como el dispositivo móvil incorpora un dispositivo GPS, quedará almacenada la posición del usuario en el momento de la petición del itinerario para los monumentos seleccionados.
5. La lista de monumentos a visitar llega al planificador mediante el módulo de comunicaciones.
6. El planificador interactúa con el módulo de gestión de transportes, que calcula los tiempos de llegada entre monumentos evaluando los distintos transportes disponibles. Ya se tiene creado un problema de planificación.
7. EL planificador obtiene la mejor solución para visitar los monumentos y la envía al dispositivo móvil.
8. El módulo de geolocalización y mapas habrá recibido por parte del módulo de comunicaciones la lista de monumentos a visitar, la procesa y obtiene de Internet sus mapas y geolocalizaciones.
9. El módulo de mapas y geolocalizaciones envía al módulo de comunicaciones los distintos

mapas y éste a su vez envía la ubicación de los mapas a los dispositivos móviles.

10. El dispositivo móvil muestra los itinerarios calculados por el planificador y los mapas obtenidos del módulo de mapas.
11. El usuario del dispositivo móvil podrá volver a realizar modificaciones volviendo a empezar el proceso descrito hasta ahora.

Este sería el funcionamiento inicial, con el paso del tiepo el dispositivo móvil irá enviando información adicional, correspondiente a la situación del usuario y el grado del cumplimiento del plan para poder tener la opción de replanificar en el caso de que ocurriera algún imprevisto.

En la siguiente página se puede encontrar una figura del sistema en general indicando los puntos o acciones anteriormente detallados. Cada una de las 11 acciones descritas anteriormente se identifica en el la figura mediante una flecha junto con el número de acción.

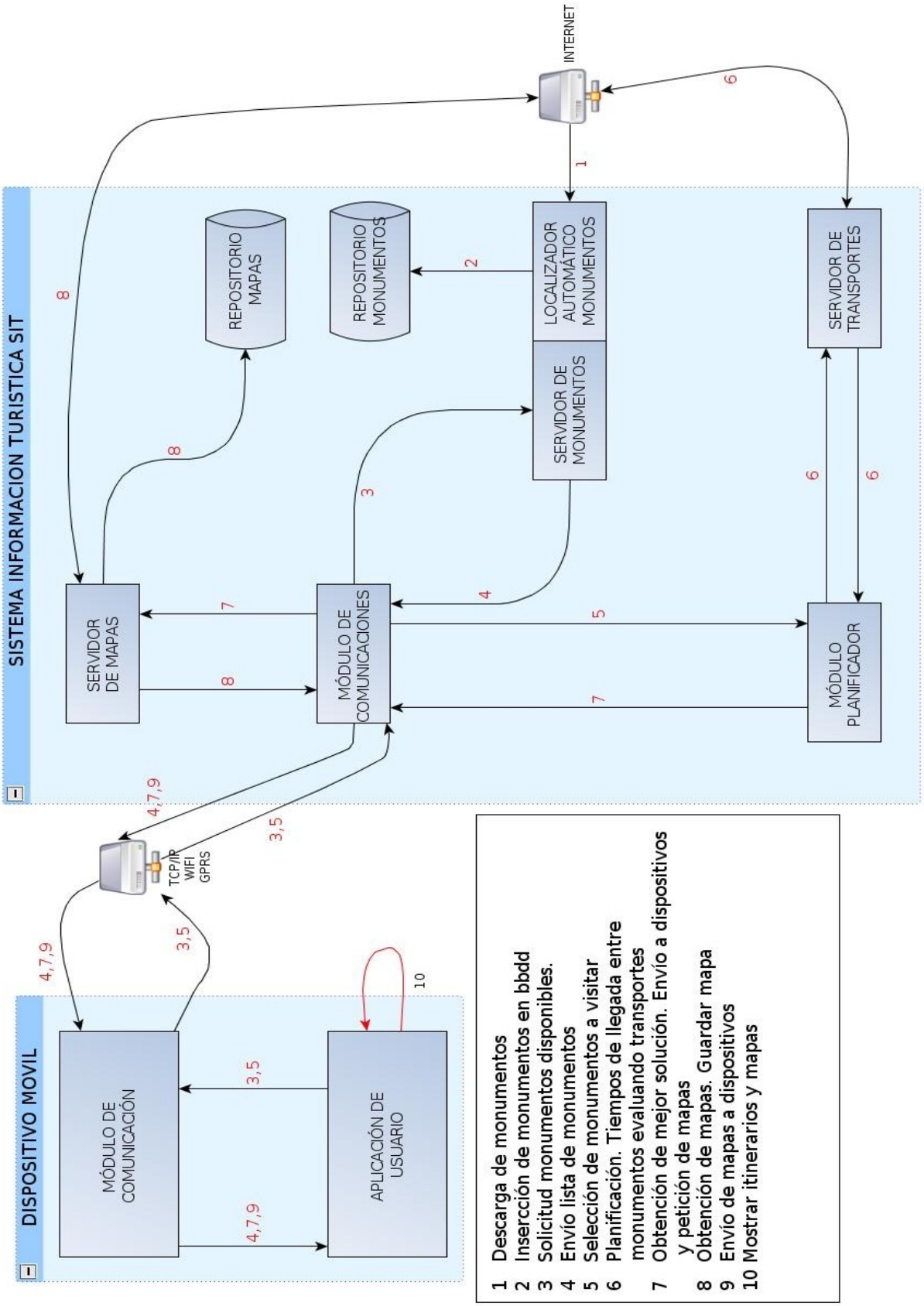

Figura 1: Diagrama del Sistema de Información Turística

1.2 Objetivos

El objetivo principal del presente PFC es la implementación y puesta en marcha de un módulo de geolocalización y gestión de mapas, de aquí en adelante MGM, englobado como un engranaje más de un Sistema de Información Turística, en adelante SIT ya descrito en el punto anterior.

Es necesario el cumplimiento de una serie de objetivos secundarios que a continuación se enumeran:

- El MGM tiene el cometido de atender las peticiones canalizadas a través del módulo de comunicaciones del SIT, iniciadas desde de los dispositivos móviles, procesarlas para poder obtener la geolocalización y el mapa y finalmente, enviar la respuesta con la localización física del mapa correspondiente a cada solicitud al módulo de comunicaciones de SIT, para que éste las reenvíe a los dispositivos móviles.
- Los mapas deben ser descargados desde algún servidor remoto, como por ejemplo yahoo maps o google maps, y almacenados en algún repositorio de datos del MGM. Es importante recalcalcar el hecho de que los mapas deben ser descargados y luego servidos directamente desde el SIT y no desde los servidores remotos de mapas (por ejemplo yahoo maps, google maps).
- El formato de intercambio de mensajes, entre el MGM y el módulo de comunicaciones del del SIT, debe utilizar tecnologías estándar como por ejemplo el lenguaje XML.[2]
- Dado que el MGM y el SIT pueden residir en distintas o la misma máquina, deben emplearse protocolos de comunicación estándar a nivel de enlace de datos, tales como Ethernet, WIFI, GPRS, HDSPA.
- Debido a que el SIT utiliza un módulo de comunicaciones que está implementado utilizando la tecnología de sockets, el MGM debe ser capaz de interconectarse con el SIT mediante esta tecnología, utilizando cualquiera de los protocolos anteriormente descritos.
- La comunicación entre el MGM y los dispositivos móviles del SIT, debe ser lo más fluida, flexible y fiable posible, con el objetivo de poder agilizar las comunicaciones priorizando el menor uso posible de ancho de banda, reduciendo de esta manera los costes económicos, derivados del uso de protocolos de comunicación con coste asociado como pueda ser GPRS, HDSPA, etc

- El MGM tiene que poder ser escalable con el objetivo de dar servicio en un futuro a más peticiones provenientes de los dispositivos móviles del SIT. Es fácil que un servicio en el mundo de las redes de comunicaciones, muera ante una avalancha de peticiones por no estar correctamente dimensionado e implementado, para que la escalabilidad pueda serle aplicada.
- El MGM debe ser capaz de tolerar solicitudes de información erróneas, bien por contenido mal formado o por fallo de conexiones a nivel de comunicaciones.
- El MGM debe ser un módulo enfocado a estar en continuo crecimiento pudiendo, en un futuro, dotarlo de diferentes servicios para satisfacer funcionalidades no soportadas actualmente. Es en este punto donde un diseño e implementación modular por parte del analista y el programador toma una importancia capital.

1.3 Organización de la memoria

El presente documento, describe y detalla la realización del proyecto final de carrera, estructurándolo en una serie de capítulos o apartados:

- **Capitulo 1: Introducción**

  Descripción general del proyecto y el sistema en el que se englobará, los objetivos a realizar y la estructura en la que se divide la memoria.

- **Capítulo 2: Estado del arte**

  Descripción general de las tecnologías actuales relacionadas con el presente proyecto final de carrera.

- **Capítulo 3: Tecnologías y herramientas utilizadas**

  En este capítulo se detallarán las tecnologías y herramientas elegidas, especificadas en en capítulo anterior. Se expondrán las razones de su elección.

- **Capitulo 4: Valoración del proyecto**

  Desglose y contabilización de los recursos necesarios para la realización del proyecto. Estimación de costes de recursos en personas y máquinas por cada una de las fases en las que se divide el proyecto.

- **Capítulo 5: Gestión del Proyecto**

  En este capítulo se detallará la metodología empleada en el análisis e implementación de los módulos en los que se divide el proyecto. Como se aborda cada uno de las partes del proyecto para que la consecución de los objetivos pueda llevarse a cabo de la forma más eficiente posible.

- **Capítulo 6: Resultados**

  Implementación de pruebas de rendimiento y carga del sistema. Evaluación de resultados en distintas situaciones reales aplicables al sistema.

- **Capítulo 7: Escalabilidad**

  Explicación de cómo se ha diseñado el sistema, para que ante una futura demanda de nuevos usuarios y conexiones, el sistema pueda escalarse y evolucionar de manera que el impacto total en rendimiento y costes sea mínimo.

- **Capítulo 8: Futuras líneas de trabajo**

  Este capítulo describe posibles funcionalidades complementarias a tener en cuenta en un futuro para poder dotar de un valor añadido al sistema actual.

- **Capítulo 9:Manual de Usuario**

  Manual de usuario para la instalación y puesta en marcha del sistema. Entorno, sistema operativo, lenguaje de programación, sistema de ficheros y estructura, servicios web y servidores web, etc.

- **Capítulo 10; Referencias**

  Índice de referencias usadas en el presente proyecto final de carrera.

2. ESTADO DEL ARTE

2.1 Descripción

En este apartado se especifica desde una visión general los temas relacionados con la implementación y realización del presente proyecto final de carrera. Los temas que se van a tratar son:

- Arquitectura basada en cliente/servidor.
- Internet y el protocolo TCP/IP.
- Introducción a los servicios web.
- ¿Qué es el software libre?
- Software libre y software privativo.
- Tecnologías y herramientas evaluadas

2.2 Arquitectura cliente / servidor

2.2.1 Visión global

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro un ámbito.

Para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y de que la arquitectura cliente/servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

Un cliente es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse

en múltiples requerimientos de trabajo a través de redes LAN[3] o WAN[4]. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Un servidor es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes tales como servicios web, impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

Técnicamente, con respecto a la definición de arquitectura cliente/servidor se encuentran las siguientes definiciones:

- Cualquier combinación de sistemas que pueden colaborar entre si para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde esta ubicada.
- Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide servicios a otro.
- Es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.
- El término cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una aplicación y un servicio soportante.
- IBM[5] define al modelo cliente/servidor como la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo en múltiples plataformas. El modelo soporta un medio ambiente distribuido, en el cual, los requerimientos de servicio hechos por estaciones de trabajo inteligentes o clientes, resultan en un trabajo realizado por otros computadores llamados servidores.
- Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

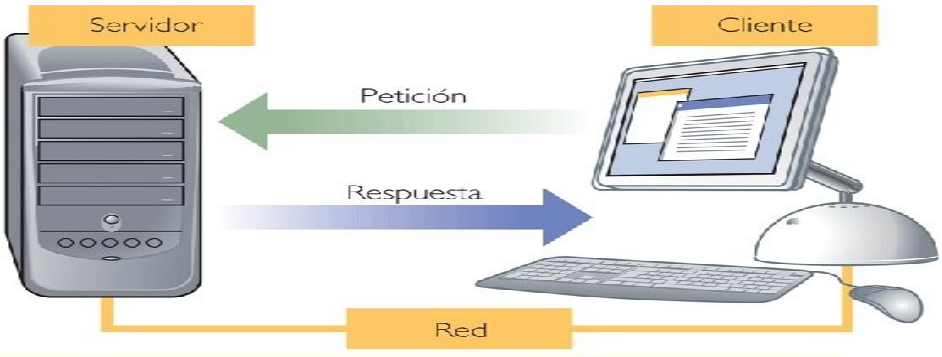

Figura 2: esquema cliente/servidor

En resumen la arquitectura cliente/servidor se puede considerar como una relación entre procesos corriendo en máquinas separadas. El servidor es un proveedor de servicios. El cliente es un consumidor de servicios.

Cliente y Servidor interactúan por un mecanismo de pasaje de mensajes, los llamados pedido de servicio y respuesta de servicios

2.2.2 Algunos antecedentes. ¿Por qué fue creado?

Existen diversos puntos de vista sobre la manera en que debería efectuarse el procesamiento de datos, aunque la mayoría coincide en que nos encontramos en medio de un proceso de evolución que se prolongará todavía por algunos años y que cambiará la forma en que se obtiene y se utiliza la información almacenada electrónicamente.

El principal motivo detrás de esta evolución, es la necesidad que tienen las organizaciones (empresas o instituciones públicas o privadas), de realizar sus operaciones más ágil y eficientemente, debido a la creciente presión competitiva a la que están sometidas. Esto se traduce en la necesidad de que su personal sea mas productivo, que se reduzcan los costes y los gastos de operación, al mismo tiempo que se generan productos y servicios más rápidamente y con mejor calidad.

En este contexto, es necesario establecer una infraestructura de procesamiento de información, que cuente con los elementos requeridos para proveer la información adecuada, exacta y oportuna en la toma de decisiones y para proporcionar un mejor servicio a los clientes.

El modelo cliente/servidor reúne las características necesarias para proveer esta infraestructura, independientemente del tamaño y complejidad de las operaciones de las organizaciones públicas o

privadas y, consecuentemente desempeña un papel importante en este proceso en evolución.

2.2.3 Evolución de la arquitectura cliente/servidor

La era de la computadora central

Desde sus inicios el modelo de administración de datos a través de computadoras se basaba en el uso de terminales remotas, que se conectaban de manera directa a una computadora central. Dicha computadora central se encargaba de prestar servicios donde cada uno de ellos se prestaba solo a un grupo exclusivo de usuarios.

La era de las computadoras dedicadas

Esta es la era en la que cada servicio empleaba su propia computadora que permitía que los usuarios de ese servicio se conectaran directamente. Esto es consecuencia de la aparición de computadoras pequeñas, de fácil uso, más baratas y más poderosas de las convencionales.

La era de la conexión libre

Hace mas de 20 años que las computadoras de escritorio aparecieron de manera masiva. Esto permitió que parte apreciable de la carga de trabajo de cómputo tanto en el ámbito de cálculo como en el ámbito de la presentación se lleven a cabo desde el escritorio del usuario. En muchos de los casos el usuario obtiene la información que necesita de alguna computadora de servicio. Estas computadoras de escritorio se conectan a las computadoras de servicio empleando software que permite la emulación de algún tipo de terminal. En otros de los casos se les transfiere la información haciendo uso de recursos magnéticos o por trascripción.

La era del cómputo a través de redes

Esta es la era que esta basada en el concepto de redes de computadoras, en la que la información reside en una o varias computadoras, los usuarios de esta información hacen uso de computadoras para trabajar y todas ellas se encuentran conectadas entre si. Esto brinda la posibilidad de que todos los usuarios puedan acceder a la información de todas las computadoras y a la vez que los diversos sistemas intercambien información.

La era de la arquitectura cliente servidor

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información,

conocidas como servidores. Estos últimos responden a la demanda del cliente que la produjo.

Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es Internet.

Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o remotazs y de procesarla como le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

2.2.4 Conclusiones finales de la arquitectura cliente/servidor

En el modelo cliente/servidor se puede encontrar las siguientes características:

- ✓ El cliente y el servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
- ✓ Las funciones de cliente y servidor pueden estar en plataformas separadas, o en la misma plataforma.
- ✓ Un servidor da servicio a múltiples clientes de forma concurrente.
- ✓ Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- ✓ La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
- ✓ Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.
- ✓ También es importante hacer notar que las funciones cliente/servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.
- ✓ Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro

de una arquitectura informática descentralizada y heterogénea.

- ✓ Además se constituye como el nexo de unión mas adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.
- ✓ Designa un modelo de construcción de sistemas informáticos de carácter distribuido.
- ✓ Su representación típica es un centro de trabajo (computadora), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los recursos de este servidor central y otros sistemas que la organización ponen a su servicio.

En conclusión, cliente/servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y, funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares.

2.3 Internet y el protocolo TCP/IP

2.3.1 Internet. Conceptos básicos y terminología

Internet es una gran red internacional de computadoras y múltiples dispositivos, o dicho de otra forma, una red de redes. Permite, como todas las redes, compartir recursos tales como el establecimiento de una comunicación inmediata con cualquier parte del mundo para obtener información sobre cualquier tema, consultar los fondos y colecciones de información de organismos públicos y privados, o conseguir un programa o cualquier otro software a ejecutar en las computadoras.

En definitiva: establecer vínculos comunicativos con millones de personas de todo el mundo, bien sea para fines académicos, de investigación o personales.

2.3.1.a Introducción

Las redes se han convertido en una parte fundamental, si no la más importante, de los actuales

sistemas de información. Constituyen el pilar en el uso compartido de la información en empresas así como en grupos gubernamentales, científicos, y en hogares y dispositivos móviles. Esta información puede adoptar distintas formas, sea como documentos, datos a ser procesados por otro ordenador, ficheros enviados a amigos, e incluso ficheros de datos con contenido de audio y vídeo.

La mayoría de estas redes se instalaron a finales de los años 60 y 70, cuando el diseño de redes se consideraba como la piedra filosofal de la investigación informática y la tecnología punta. Dio lugar a numerosos modelos de redes como la tecnología de conmutación de paquetes, redes de área local con detección de colisión, redes jerárquicas en empresas, y muchas otras de elevada calidad.

Desde comienzos de los años 70, otro aspecto de la tecnología de redes cobró importancia: el modelo de pila de protocolo, que permite la interoperabilidad entre aplicaciones. Toda una gama de arquitecturas fue propuesta e implementada por diversos equipos de investigación y fabricantes de ordenadores.

El resultado de todos estos conocimientos tan prácticos es que hoy en día cualquier grupo de usuarios puede hallar una red física y una arquitectura adecuada a sus necesidades específicas, desde líneas asíncronas de bajo coste, sin otro método de recuperación de errores que una función de paridad bit a bit, pasando por funciones completas de redes de área extensa(pública o privada) con protocolos fiables como redes públicas de conmutación de paquetes o redes privadas, hasta las redes de área local, de alta velocidad pero distancia limitada.

El lado negativo de esta explosión de la información es la penosa situación que se produce cuando un grupo de usuarios desea extender su sistema informático a otro grupo de usuarios, que resulta que tiene una tecnología y unos protocolos de red diferentes. En consecuencia, aunque pudieran ponerse de acuerdo en el tipo de tecnología de red para conectar físicamente sus instalaciones, las aplicaciones, como por ejemplo sistemas de correo, serían aún incapaces de comunicarse entre sí debido a los diferentes protocolos.

Se tomó conciencia de esta situación bastante temprano (a comienzo de los años 70), gracias a un grupo de investigadores en los Estados Unidos, que fueron artífices de un nuevo paradigma: la interconexión de redes. Otras organizaciones oficiales se implicaron en la interconexión de redes, tales como ITU-T[6] e ISO[7]. Todas trataban de definir un conjunto de protocolos, distribuidos en un conjunto bien definido de capas, de modo que las aplicaciones pudieran comunicarse entre sí, con independencia de la tecnología de red subyacente y del sistema operativo sobre el que se ejecutaba cada aplicación.

2.3.1.b Internet

¿Qué es exactamente? En primer lugar, la palabra Internet es simplemente una contracción de la frase red interconectada. Sin embargo, escrita con mayúscula hace referencia a un conjunto mundial de redes interconectadas, de tal forma que Internet es una red interconectada, aunque no a la inversa. A Internet se le llama a veces "Interred conectada"("connected Internet"). Internet está constituida por los siguientes grupos de redes:

- Troncales: grandes redes que existen principalmente para interconectar otras redes. Actualmente las redes troncales son NSFNET en US, EBONE en Europa y las grandes redes troncales comerciales.
- Redes regionales que conectan, por ejemplo, universidades y colegios.
- Redes comerciales que suministran acceso a troncales y suscriptores, y redes propiedad de organizaciones comerciales para uso interno que también tienen conexión con Internet.
- Redes locales, como por ejemplo, redes a nivel de campus universitario.
- En muchos casos, particularmente en redes de tipo comercial, militar y gubernamental, el tráfico entre estas y el resto de Internet está restringido por ejemplo mediante cortafuegos.

Esto conduce a la pregunta ¿Cómo sé si estoy conectado a Internet? Un enfoque viable es preguntarse: ¿puedo hacer un ping al host uc3m.es ? El ping es un programa usado para determinar si un host de una red es alcanzable; está implementado en cualquier plataforma TCP/IP. Si la respuesta es no, entonces no estás conectado. Esta definición no implica necesariamente que uno esté totalmente aislado de Internet: muchos sistemas que fallarían en este test tienen, por ejemplo, pasarelas de correo electrónico a Internet.

2.3.2 TCP/IP

2.3.2.a Internetworking

La pila TCP/IP se llama así por dos de sus protocolos más importantes: TCP ("Transmission Control Protocol") de IP ("Internet Protocol"). Otro nombre es pila de protocolos de Internet, y es la frase oficial usada en documentos oficiales de estándares. En este documento utilizaremos el término TCP/IP, que es más habitual.

La primera meta de diseño de TCP/IP fue construir una interconexión de redes que proporcionase servicios de comunicación universales: una red, o internet. Cada red física tiene su propia interfaz de comunicaciones dependiente de la tecnología que la implementa, en la forma de una interfaz de programación que proporciona funciones básicas de comunicación(primitivas).

Las comunicaciones entre servicios las proporciona el software que se ejecuta entre la red física y la aplicación de usuario, y da a estas aplicaciones una interfaz común, independiente de la estructura de la red física subyacente. La arquitectura de las redes físicas es transparente al usuario.

El segundo objetivo es interconectar distintas redes físicas para formar lo que al usuario le parece una única y gran red. Tal conjunto de redes interconectadas se denomina "internetwork" o internet. Para poder interconectar dos redes, necesitamos un ordenador que esté conectado a ambas redes y que pueda retransmitir paquetes de una a la otra; tal máquina es un "router". El término "router IP" también se usa porque la función de encaminamiento es parte de la capa IP de la pila TCP/IP. La siguiente figura muestra dos ejemplo de "internetworks".

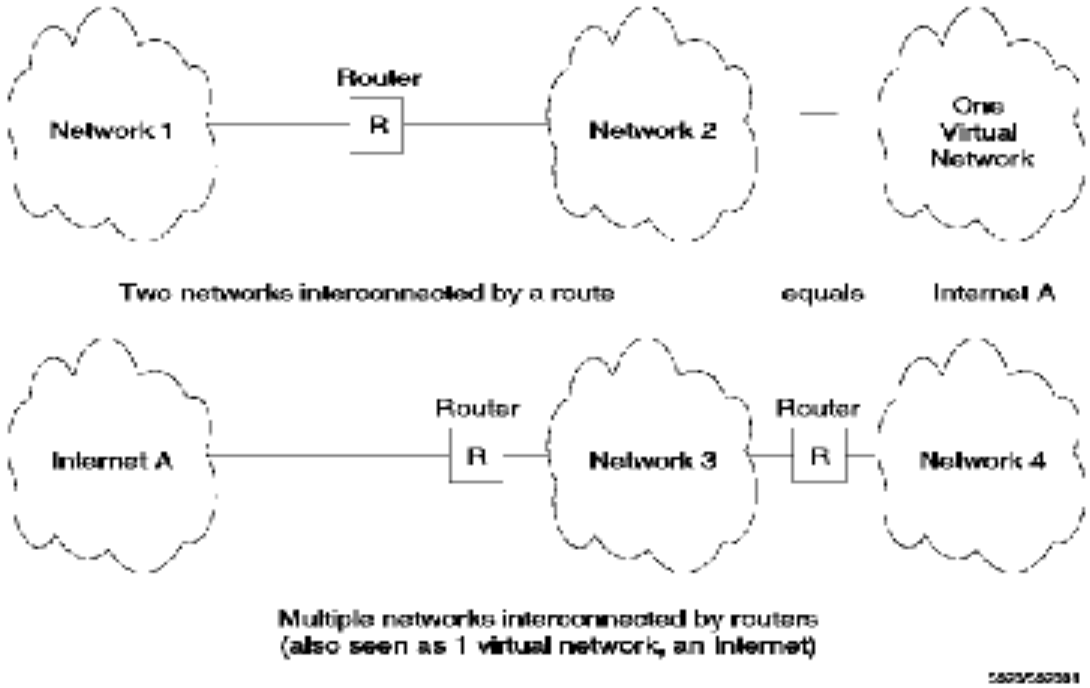

Figura 3: Dos conjuntos interconectados de redes, cada uno visto como una red lógica

Las propiedades básicas de un "router" son:

- Desde el punto de vista de la red, es un host normal.
- Desde el punto de vista del usuario, es invisible. El usuario sólo ve una gran red.

Para ser capaz de identificar un host en la red, a cada se le asigna una dirección, la *dirección IP*. Cuando un host tiene múltiples adaptadores de red, cada adaptador tiene una dirección IP separada. La dirección IP consta de dos partes:

dirección IP = <número de red><número de host>

El número de red lo asigna una autoridad central y es unívoco en Internet. La autoridad para asignar el número de host reside en la organización que controla la red identificada por el número de red

2.3.2.b Ventajas

El conjunto TCP/IP está diseñado para enrutar y tiene un grado muy elevado de fiabilidad, es adecuado para redes grandes y medianas, así como en redes empresariales. Se utiliza a nivel mundial para conectarse a Internet y a los servidores web. Es compatible con las herramientas estándar para analizar el funcionamiento de la red.

Es rápido en redes con un volumen de tráfico grande donde haya que enrutar un gran número de tramas.

El conjunto TCP/IP se utiliza tanto en redes empresariales como por ejemplo en campus universitarios o en complejos empresariales, en donde utilizan muchos enrutadores y conexiones a mainframe o a ordenadores UNIX, como así también en redes pequeñas o domésticas, y hasta en teléfonos móviles y en domótica.

2.4 Introducción a los servicios web (web services)

Si se realiza una búsqueda de los términos "servicio web" en el buscador Google, sale como resultado más de 13 millones de páginas.

Evidentemente no todas hablan sobre el tema, pero puede decirse que los servicios web son una de las tecnologías más atractivas del mundo internet.

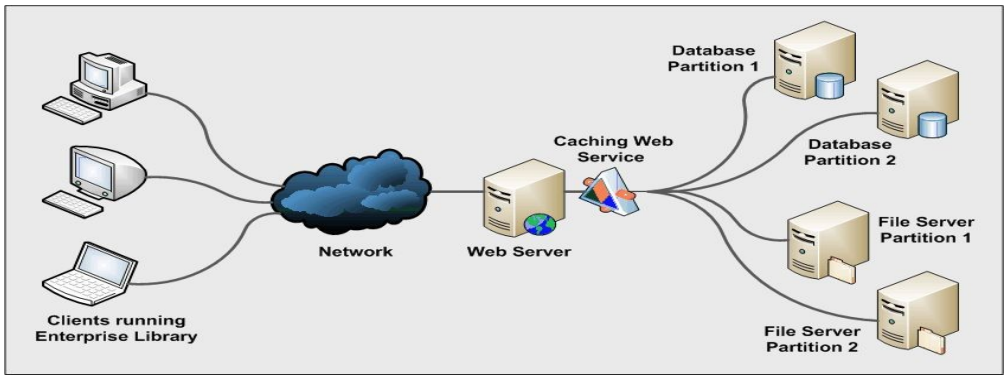

Figura 4: ejemplo de servicio web

Hace ya algún tiempo que esta tecnología dejó de ser una moda y una promesa, para convertirse en una realidad con muchas utilidades. Google es un claro ejemplo de ello. Esta compañía inició en Abril del año 2002 la posibilidad de hacer búsquedas usando su motor de búsqueda (que indexa más de 3.000 millones de documentos Web). Esto se hace mediante servicio web. Como se puede ver, una aplicación real muy utilizada hoy en día.

Pero, ¿qué es un servicio web?

2.4.1 Definición de la W3C

Según la W3C[8] (el organismo que se encarga de desarrollar gran parte de los estándares de internet), se define un servicio web de la siguiente forma: "Un servicio web es una aplicación software

identificada mediante una URI[9], cuyo interfaz y uso es capaz de ser definido, descrito y descubierto mediante artefactos XML, y soportar interacciones directas con otras aplicaciones software usando mensajes basados en XML y protocolos basados en Internet".

Una definición alternativa podría ser la siguiente: Un servicio web es un componente software que se basa en las siguientes tecnologías:

- Un formato que describa la interfaz del componente (sus métodos y atributos) basado en XML. Por lo general este formato es el WSDL[10].
- Un protocolo de aplicación basado en mensajes y que permite que una aplicación interaccione, use, instancie, llame o ejecute al servicio web. Por lo general este protocolo es SOAP[11] .
- Un protocolo de transporte que se encargue de transportar los mensajes por internet. Por lo general este protocolo de transporte es HTTP[12] que es exactamente el mismo que se usa para navegar por la Web.

2.4.2 ¿Qué son realmente?

Los servicios web, no son por tanto aplicaciones con una interfaz gráfica con los cuales las personas puedan interaccionar, sino que son software accesible por otras aplicaciones en internet o en redes privadas que usan tecnologías internet. De esta forma se puede desarrollar aplicaciones que hagan uso de otras aplicaciones que estén disponibles en internet interaccionando con ellas.

Un ejemplo podría ser una pequeña parte del presente proyecto final de carrera, un servicio web al que se le pudiese preguntar por una ubicación terrestre y devuelva los datos geolocalizados de dicha ubicación.

De esta forma cualquier aplicación, ya sea web o de escritorio, que quiera mostrar o usar esta información sólo tendría que solicitarla a través de internet al servicio web cuando la necesitase.

Otro ejemplo de servicio web podría ser uno que al pasarle el nombre de una ciudad, devolviese la temperatura, humedad, y otras condiciones climatológicas de la misma.

Los servicios web no son la panacea, sino una tecnología apropiada para resolver ciertos problemas. Básicamente los servicios web permiten que diferentes aplicaciones, realizadas con diferentes tecnologías, y ejecutándose en toda una variedad de entornos, puedan comunicarse e integrarse, lo cual es muy importante, y ¿por qué?.

El coste de desarrollar software siempre ha sido altísimo, y la diversidad de las plataformas una realidad desde el inicio de la informática. De hecho a medida que aumentaba la complejidad de las aplicaciones que las empresas demandaban, aumentaba considerablemente el gasto económico en desarrollarlas. Para enfrentarse a esta situación se han seguido diferentes líneas todas ellas encaminadas a reutilizar las aplicaciones ya desarrolladas.

Una de estas propuestas es la estandarización de lenguajes de programación de tal forma que si cualquiera escribía un programa en C, solo se necesitaba un compilador de C en la plataforma específica en la que se fuese a ejecutar la aplicación. Esto en la realidad es difícil de hacer funcionar, porque en seguida surgieron pequeñas diferencias y extensiones de C que hacían difícil 'transportar' una aplicación entre diferentes plataformas.

Otra posibilidad ha sido la que ha desarrollado la empresa Sun con Java. Se programa para una plataforma, pero para una plataforma virtual, en este caso para la máquina virtual Java, y en cada plataforma real se implementa una máquina virtual de Java, que será la encargada de ejecutar las aplicaciones escritas en Java. Las implementaciones son específicas de cada plataforma pero al ser todas las máquinas virtuales exactamente la misma, los programas escritos en Java deben poder ejecutarse sin ningún problema en todas las plataformas que tengan una máquina virtual de Java. Esta apuesta ha demostrado ser muy útil y funcionar bastante bien.

Sin embargo, ¿qué pasa si se tienen varias aplicaciones ya desarrolladas en lenguajes propietarios o en plataformas específicas y es necesaria interacción entre ellas?. El coste de elegir a posteriori un único lenguaje o plataforma y migrarlo al mismo es descabellado en la mayoría de las situaciones, y es aquí donde los servicios web, así como otras tecnologías, pueden ser de una grandísima utilidad.

Con los servicios web es posible reutilizar desarrollos ya utilizados sin importar la plataforma en la que funcionan o el lenguaje en el que están escritos. Los servicios web se constituyen en una capa adicional a estas aplicaciones de tal forma que pueden interaccionar entre ellas usando para ello tecnologías estándares que han sido desarrolladas en el contexto de internet.

2.4.3 Ámbitos de uso

En la actualidad existe un gran número de servicios web disponibles en Internet, con infinidad de objetivos y acciones que realizan y sobre todo utilizados en distintos ámbitos de uso:

- Uso en el ámbito académico y público en distintas disciplinas: Por ejemplo servicios dedicados a la consulta de fondos de bibliotecas, servicios web de intercambio de información entre colaboradores (universidades, centros académicos, etc)

- Uso en el ámbito de la comunicación y social: servicios web de correo electrónico, mensajería instantánea, redes sociales, etc.

- Uso en multitud de escenarios: servicios web que permiten la geolocalización, y navegación por medio de mapas, predicción del tiempo, estado de carreteras.

- Ámbito municipal, estatal y nacional : servicios web para la tramitación de expedientes y trámites administrativos a nivel de comunidad autónoma, provincia, municipio etc.

- Ámbito sanitario: Intercambio de información, tratamientos entre hospitales.

- Ámbitos de las cuerpos y organismos oficiales: Intercambio de información entre instituciones oficiales, policía, guardia civil, etc

- Ámbito económico y comercio: Tiendas especializadas en el comercio electrónico, compra venta de acciones y títulos en bolsa, uso de banca a distancia, compraventa de bienes materiales, compra de productos en grandes superficies, gestión de partes de accidente en compañías de seguros, etc

- Ámbito social: Existen multitud de servicios web para la obtención de noticias y hechos, acceso a hemerotecas de diarios, televisiones.

- Ámbito del entretenimiento: existencia de múltiples servicios para la demanda de vídeo online en tiempo real, gestión de RSS y suscripciones a canales información, juegos en red y videoconsolas.

- Ámbito de transporte y logística, consulta y seguimiento de envíos de mercancías de los proveedores logísticos, obtención de datos de vuelos y reservas de los mismos en el sector aeroportuario.

- Ámbito científico: computación distribuida, redes de colaboración científicas.

- Por no mencionar todos aquellos que poseen los ejércitos e instituciones de carácter militar para el intercambio de información y acciones que se puedan llevar a cabo de forma remota.

Como se puede observar, están muy implantados en la sociedad a nivel mundial, incluso la dependencia en ellos en algunos casos es muy grande, ya que, si por ejemplo, hoy en día el correo electrónico o el intercambio de información de reservas para viajes aéreas fallasen, ocasionaría infinidad de problemas y pérdidas económicas de forma instantánea, por no mencionar otros servicios web que puedan pueden verse afectados colateralmente.

Son por tanto en muchos casos, necesidades de la sociedad actual, que permiten que muchos mecanismos utilizados en infinidad de ámbitos, puedan funcionar eficaz y rápidamente, repercutiendo directamente en beneficios para cualquier actor (empresas, instituciones y organismos , ciudadanos) presentes en los distintos sectores en los que se utilizan o bien directa o indirectamente en los que dependen de ellos.

Muchos de estos servicios enumerados, se encuentran disponibles de forma abierta y libre de uso o por el contrario restringida a un determinado conjunto de organizaciones, estados, empresas y/o personas. Pueden estar restringidas a un cierto colectivo en base a la naturaleza de las acciones y entrono sobre el que se aplica (defensa, cuerpos y seguridad el estado, control aeroportuario) , o debido a que trate de servicios de suscripción económica.

2.5 Software libre

Según la organización GNU[13] "software libre" se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa, y adaptarlo a cualquier necesidad . El acceso al código fuente es una condición previa para esto.

- La libertad de distribuir copias, con lo que se puede ayudar a otras personas.
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Un programa es software libre si los usuarios tienen todas estas libertades. Así pues, se debe tener la libertad de distribuir copias, sea con o sin modificaciones, sea gratis o cobrando una cantidad por la distribución, a cualquiera y en cualquier lugar.

También existe la libertad de hacer modificaciones y utilizarlas de manera privada el trabajo u ocio, sin ni siquiera tener que anunciar que dichas modificaciones existen. Si se publica cualquier cambio, no es obligatorio avisar a nadie en particular, ni de ninguna manera en particular.

La libertad para usar un programa significa la libertad para cualquier persona u organización de usarlo en cualquier tipo de sistema informático, para cualquier clase de trabajo, y sin tener obligación de comunicárselo al desarrollador o a alguna otra entidad específica.

La libertad de distribuir copias debe incluir tanto las formas binarias o ejecutables del programa como su código fuente, sean versiones modificadas o sin modificar (distribuir programas de modo ejecutable es necesario para que los sistemas operativos libres sean fáciles de instalar).

Para que las libertades de hacer modificaciones y de publicar versiones mejoradas tengan sentido, es obligatorio tener acceso al código fuente del programa. Por lo tanto, la posibilidad de acceder al código fuente es una condición necesaria para el software libre.

Para que estas libertades sean reales, deben ser irrevocables mientras no se haga nada incorrecto; si el desarrollador del software tiene el poder de revocar la licencia aunque no se le haya dado motivos, el software no es libre.

Son aceptables, sin embargo, ciertos tipos de reglas sobre la manera de distribuir software libre, mientras no entren en conflicto con las libertades centrales. Por ejemplo, “copyleft” es la regla que implica que, cuando se redistribuya el programa, no se pueden agregar restricciones para denegar a otras personas las libertades centrales. Esta regla no entra en conflicto con las libertades centrales, sino que más bien las protege.

Así pues, quizás se haya pagado para obtener copias de software GNU, o tal vez se ha obtenido sin ningún coste. Pero independientemente de cómo se han conseguido las copias, siempre se tiene la libertad de copiar y modificar el software, e incluso de vender copias.

“Software libre” no significa “no comercial”. Un programa libre debe estar disponible para uso comercial, desarrollo comercial y distribución comercial. El desarrollo comercial del software libre ha dejado de ser inusual; el software comercial libre es muy importante.

Es aceptable que haya reglas acerca de cómo empaquetar una versión modificada, siempre que no bloqueen a consecuencia de ello la libertad de publicar versiones modificadas. Reglas como “si está disponible el programa de esta manera, puede hacerse disponible también de esta otra” pueden ser igualmente aceptables, bajo la misma condición. También es aceptable que la licencia requiera que, si se ha distribuido una versión modificada y el desarrollador anterior pide una copia de ella, pueda obtenerla.

En el proyecto GNU, se utiliza el término “copyleft” para proteger de modo legal estas libertades para todos. Pero el software libre sin “copyleft” también existe. Hay razones importantes por las que es mejor usar “copyleft”.

A veces las normas de control de exportación del gobierno y las sanciones mercantiles pueden restringir tu libertad de distribuir copias de los programas a nivel internacional. Los desarrolladores de software no tienen el poder de eliminar o sobrepasar estas restricciones, pero lo que pueden y deben hacer es rehusar el imponerlas como condiciones de uso del programa. De esta manera, las restricciones no afectarán a actividades y gente fuera de las jurisdicciones de estos gobiernos.

Cuando se habla de software libre, es mejor evitar términos como: “regalar” o “gratis”, porque esos términos implican que lo importante es el precio, y no la libertad.

2.6 Software libre y software privativo

El software, como producto que se comercializa, por lo general no está a la venta. Lo que el usuario adquiere, a través de una erogación monetaria o sin ella, es una licencia respecto de los usos que puede dar a los programas en cuestión. Esto es, a diferencia de por ejemplo un libro o un disco, bienes comercializados tangibles, en los que el cliente adquiere título real sobre algo que puede prestar, regalar, revender, citar, alquilar, resumir, etc.

Al comprar un programa, el usuario por regla general no adquiere derecho de propiedad alguno, en muchos casos ni siquiera pasa a ser propietario del medio magnético u óptico en el que el software es entregado, que continúa siendo propiedad del autor original.

La licencia de uso de un programa en particular regula las maneras en las que el usuario puede utilizarlo. Si bien la variedad de tipos de licencia abarca todo el rango de posibilidades, desde las condiciones más leoninas hasta las más liberales, se las puede clasificar en dos grandes categorías: por un lado están las licencias conocidas como libres, y por otro las privativas. La gran diferencia entre estos tipos de licencia consiste en que un software licenciado de modo privativo por lo general otorga al usuario solamente el derecho de ejecutar el programa "tal como es" (es decir, con errores incluidos) en determinada computadora, prohibiendo expresamente otro uso, mientras que el software gobernado por una licencia libre permite al usuario no solo ejecutar el programa en tantas computadoras como desee, sino también copiarlo, inspeccionarlo, modificarlo, mejorarlo, corregir errores y distribuirlo, o contratar a alguien para que lo haga por él.

Estos derechos adicionales son herramientas clave e indispensables de todo software que vaya a ser usado en el entorno de la administración pública.

A continuación se muestra una figura que resume de una manera muy completa la filosofía del software libre

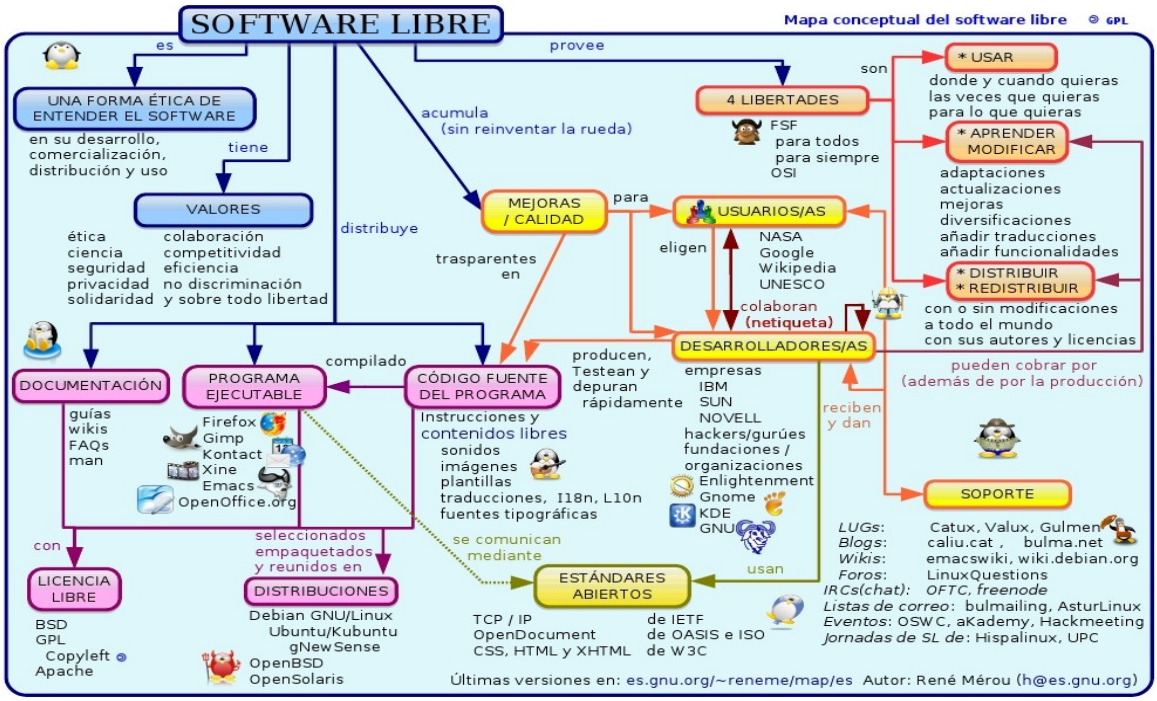

Figura 5: Características del software libre y su filosofía

2.7 Sistema operativo

En el mundo de las computadoras u ordenadores, el sistema operativo es el software esencial del mismo.

El sistema operativo establece las reglas y parámetros para que el software y las aplicaciones interactúen con la computadora, ya que en lugar de hablar directamente con el hardware, las aplicaciones hablan con el sistema operativo y este actúa como su intérprete. Si no existiera el sistema operativo, cada empresa desarrolladora de software tendría que crear su propio método para que las aplicaciones graben archivos en el disco duro, desplegar textos y gráficos en la pantalla, enviar texto a la impresora e infinidad de funciones más.

Existen multitud de sistemas operativos desarrollados sobre diversos entornos computacionales. Aún así, se establece una pequeña clasificación de sistemas operativos según la naturaleza de su uso, tales como sistemas operativos que actúan en servidores o sistemas operativos que actúan en ordenadores de escritorio o en "modo cliente".

2.7.1 Microsoft Windows

Microsoft[14] Windows es el nombre de una serie de sistemas operativos desarrollados por Microsoft

desde 1981, año en que el proyecto se denominaba "Interface Manager".

Anunciado en 1983, Microsoft comercializó por primera vez el entorno operativo denominado Windows en noviembre de 1985 como complemento para MS-DOS, en respuesta al creciente interés del mercado en una interfaz gráfica de usuario (GUI, Graphical user Interface) Microsoft Windows llegó a dominar el mercado de ordenadores personales del mundo, superando a Mac OS, el cual había sido introducido previamente a Windows. En octubre de 2009, Windows tenía aproximadamente el 91% de la cuota de mercado de sistemas operativos en equipos cliente que acceden a Internet. Las versiones más recientes de Windows son Windows 7 para equipos de escritorio, Windows Server 2008 R2 para servidores y Windows Mobile 7 para dispositivos móviles.

Microsoft también posee diferentes versiones de sus sistema operativo enfocado al ámbito de servidores. Desde Windows NT, pasando por Windows 2000 Server hasta el último Windows Server 2008, Microsoft ha desarrollado estos sistemas con modificaciones especiales para poder ejecutarse como servidores y proporcionar a las empresas y usuarios software y aplicaciones que permitan realizar las tareas más importantes necesarias en el mundo de las redes, como gestión de correo, servidor web y bases de datos, gestión de perfiles de usuarios etc.

2.7.1.a Críticas y problemas

Windows, ya desde sus inicios, ha estado envuelto en la polémica. Al principio se decía que Windows era una copia del sistema operativo de Apple; más adelante se hablaba de si existía competencia desleal con algunos programas que se incluían dentro del sistema. Con la aparición del software libre las polémicas se orientan a la política de código cerrado de Microsoft.

Las mayores críticas que recibió Windows hasta la versión Windows XP Service Pack 2 era la estabilidad del sistema, el sistema operativo presentaba varios fallos de distinta índole y gravedad, los cuales fueron disminuyendo con el correr de las versiones. Desde Microsoft siempre expresaron que estos fallos se debían a aplicaciones externas a Windows, pero algunos fallos se producían apenas instalado el sistema, sin siquiera haber agregado programa alguno.

Microsoft ha lanzado una campaña, llamada "Get the facts" en la que muestra cientos de empresas conocidas que migraron de GNU/Linux a Windows Server y aumentaron su productividad. Los defensores de GNU/Linux desarrollaron su propio estudio argumentando que, en contra de uno de los reclamos de Microsoft, GNU/Linux tiene menores costos administrativos que Windows

Server.Otro estudio realizado por el Yankee Group afirma que la actualización desde una versión de Windows Server a otra plataforma tiene un coste inferior al de cambiar de GNU/Linux a Windows Server.

2.7.1.b Seguridad

Una de las principales críticas que con frecuencia reciben los sistemas operativos Windows es la debilidad del sistema en lo que a seguridad se refiere y el alto índice de vulnerabilidades críticas. El propio Bill Gates, fundador de Microsoft, ha asegurado en repetidas ocasiones que la seguridad es objetivo primordial para su empresa.

Partiendo de la base de que no existe un sistema completamente libre de errores, las críticas se centran en la lentitud con la que la empresa reacciona ante un problema de seguridad que pueden llegar a meses o incluso años de diferencia desde que se avisa de la vulnerabilidad hasta que se publica un parche.

En algunos casos la falta de respuesta por parte de Microsoft ha provocado que se desarrollen parches que arreglan problemas de seguridad hechos por terceros.

Uno de los pilares en que se basa la seguridad de los productos Windows es la seguridad por ocultación, en general, un aspecto característico del software propietario que sin embargo parece ser uno de los responsables de la debilidad de este sistema operativo ya que, la propia seguridad por ocultación, constituye una infracción del principio de Kerckhoff, el cual afirma que la seguridad de un sistema reside en su diseño y no en una supuesta ignorancia del diseño por parte del atacante

2.7.2 Apple Inc

Apple15 Inc. es una empresa multinacional estadounidense con sede en Cupertino, California, que diseña y produce equipos electrónicos y software. Entre el software que desarrolla Apple se encuentran el sistema operativo Mac OS X.

2.7.2.a Mac OSX

Mac OS X es la décima versión del sistema operativo de Apple para computadores Macintosh. Las versiones previas usaron una numeración cardinal, p.j. Mac OS 8 y Mac OS 9. La letra X en el nombre Mac OS X se refiere al 10 en números romanos. Por tal motivo, la pronunciación correcta es "diez" en este contexto, aunque pronunciarlo como "equis" es muy común. El núcleo del Mac OS X es compatible con POSIX construido sobre el núcleo XNU, con facilidades UNIX disponibles en

la interfaz de línea de comandos (terminal). Apple liberó esta familia de software como un sistema operativo libre y de código abierto, bajo el nombre de Darwin, pero parcialmente se fue volviendo código cerrado. Sobre Darwin, Apple colocó varios componentes, incluyendo la interfaz de usuario Aqua y el Finder, para completar la interfaz en la que estaba basado Mac OS X.16

Mac OS X introdujo un buen número de nuevas funciones para proveer una plataforma más viable y estable que su predecesora, el Mac OS 9. Por ejemplo, la multitarea preventiva y la memoria protegida mejoraron la habilidad del sistema para ejecutar múltiples aplicaciones simultáneamente sin interrupciones.

Existe una versión de Mac OS X para servidores, llamada Mac OS X Server. Es idéntico a su versión de escritorio, pero incluye además herramientas administrativas gráficas para la gestión de usuarios, redes, y servicios de red como LDAP, Servidor de correo, Servidor Samba, DNS, entre otros. También incorpora en sus versiones más recientes un número adicional de servicios y herramientas para configurarlos, tales como Servidor web, herramientas para crear una Wiki, Servidor iChat, etc.

2.7.3 GNU Linux

Linux es un sistema operativo libre tipo Unix. Es usualmente utilizado junto a las herramientas GNU como interfaz entre los dispositivos de hardware y los programas usados por el usuario para manejar un computador. A la unión de ambas tecnologías, más la inclusión de algunas otras, (como entornos de escritorio e interfaces gráficas) se le conoce como distribución GNU/Linux. Fue lanzado bajo la licencia pública general de GNU y es desarrollado gracias a contribuciones provenientes de colaboradores de todo el mundo, por lo que es uno de los ejemplos más notables de software libre. Debido a su naturaleza de contenido libre, ambos proyectos invitan a colaborar en ellos de forma altruista.

Linux fue creado por Linus Torvalds en 1991. Muy pronto, la comunidad de Minix (un clon de Unix), contribuyó en el código y en ideas para el núcleo Linux. Por aquel entonces, el Proyecto GNU ya había creado muchos de los componentes necesarios para conseguir un entorno operador con software libre, pero su propio sistema operativo, el llamado (GNU Hurd), se encontraba incompleto por lo que comenzaron a usar Linux. El día en que el proyecto GNU estime que Hurd es suficiente robusto y estable, será llamado a reemplazar a Linux. Es por esto que a pesar de las funcionalidades limitadas de la primera versión, rápidamente Linux fue acumulando desarrolladores

y usuarios que adoptaron el código de estos proyectos para usar con el nuevo sistema operativo. Hoy en día el núcleo Linux ha recibido contribuciones de miles de programadores. Linux es el tercer sistema operativo más utilizado en el entorno de escritorio (desktop) y el más utilizado en el entorno de servidores. Actualmente posee una cuota de mercado del 1,02% a nivel mundial.

2.7.3.a Arquitectura

Actualmente Linux es un núcleo monolítico híbrido. Los controladores de dispositivos y las extensiones del núcleo normalmente se ejecutan en un espacio privilegiado conocido como anillo 0 (ring 0), con acceso ilimitado al hardware, aunque algunos se ejecutan en espacio de usuario. A diferencia de los núcleos monolíticos tradicionales, los controladores de dispositivos y las extensiones al núcleo se pueden cargar y descargar fácilmente como módulos, mientras el sistema continúa funcionando sin interrupciones. También, a diferencia de los núcleos monolíticos tradicionales, los controladores pueden ser prevolcados (detenidos momentáneamente por actividades más importantes) bajo ciertas condiciones. Esta habilidad fue agregada para gestionar correctamente interrupciones de hardware, y para mejorar el soporte de multiprocesamiento simétrico.

El hecho de que Linux no fuera desarrollado siguiendo el diseño de un micronúcleo (diseño que, en aquella época, era considerado el más apropiado para un núcleo por muchos teóricos informáticos) fue asunto de una famosa y acalorada discusión entre Linus Torvalds y Andrew S. Tanenbaum.

A diferencia de los núcleos monolíticos tradicionales, los controladores de dispositivos son fácilmente configurables como módulos del núcleo cargables, y se pueden cargar o descargar mientras se está ejecutando el sistema.

2.7.3.b Lenguajes de programación

Linux está escrito en el lenguaje de programación C, en la variante utilizada por el compilador GCC (que ha introducido un número de extensiones y cambios al C estándar), junto a unas pequeñas secciones de código escritas con el lenguaje Ensamblador. Por el uso de sus extensiones al lenguaje, GCC fue durante mucho tiempo el único compilador capaz de construir correctamente Linux. Sin embargo, Intel afirmó haber modificado su compilador C de forma de poder compilarlo correctamente.

Asimismo se usan muchos otros lenguajes en alguna forma, básicamente en la conexión con el proceso de construcción del núcleo (el método a través del cual las imágenes boteables son creadas

desde el código fuente). Estos incluyen a Perl, Python y varios lenguajes shell scripting. Algunos drivers también pueden ser escritos en C++, Fortran, u otros lenguajes, pero esto no es aconsejable. El sistema de construcción de Linux oficialmente solo soporta GCC como núcleo y compilador de controlador.

2.7.3.c Portabilidad

Aun cuando Linus Torvalds no ideó originalmente Linux como un núcleo portable, ha evolucionado en esa dirección. Linux es ahora de hecho, uno de los sistemas operativos más ampliamente portados, y funciona en sistemas muy diversos que van desde iPAQ (una handheld) hasta un zSeries (un mainframe masivo). Está planeado que Linux sea el sistema operativo principal de las nuevas supercomputadoras de IBM, Blue Gene cuando su desarrollo se complete.

De todos modos, es importante notar que los esfuerzos de Torvalds también estaban dirigidos a un tipo diferente de portabilidad. Según su punto de vista, la portabilidad es la habilidad de compilar fácilmente en un sistema aplicaciones de los orígenes más diversos; así, la popularidad original de Linux se debió en parte al poco esfuerzo necesario para tener funcionando las aplicaciones favoritas de todos, ya sean GPL o de Código abierto.

Las arquitecturas principales soportadas por Linux son DEC Alpha, ARM, AVR32, Blackfin, ETRAX CRIS, FR-V, H8, IA64, M32R, m68k, MicroBlaze, MIPS, MN10300, PA-RISC, PowerPC, System/390, SuperH, SPARC, x86, x86 64 y Xtensa

2.7.3.d Copyright

Inicialmente, Torvalds distribuyó Linux bajo los términos de una licencia que prohibía la explotación comercial. Pero esta licencia fue reemplazada, poco tiempo después, por la GNU GPL (versión 2 exclusivamente). Los términos de esta última licencia permiten la distribución y venta de copias o incluso modificaciones, pero requiere que todas las copias del trabajo original y trabajos de autoría derivados del original sean publicados bajo los mismos términos, y que el código fuente siempre pueda obtenerse por el mismo medio que el programa licenciado.

Torvalds se ha referido a haber licenciado Linux bajo la GPL como *"la mejor cosa que he hecho nunca"*.

Sin embargo, la versión oficial del núcleo Linux contiene firmware de código cerrado, por ello, el Proyecto Linux-libre, auspiciado por la FSFLA, publica y mantiene versiones modificadas del

núcleo Linux a las que se les ha quitado todo el software no libre.

2.8 Lenguajes de programación

Los lenguajes de programación son herramientas que nos permiten crear programas y software. Una computadora funciona bajo control de uno o varios programas ejecutado en un sistema operativo el cual debe estar almacenado en la unidad de memoria.

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.

El código escrito en este tipo de lenguaje se transforma en código máquina binario, para que el procesador de la computadora pueda ejecutarlo.

Existen múltiples clasificaciones de los lenguajes de programación, tales como lenguajes imperativos o funcionales, compilados o interpretados, multiplataforma o únicos, etc.

A continuación, se detallarán varios lenguajes de programación teniendo en cuenta que, el lenguaje debe poder ser usado en un ámbito de redes de computadores.

2.8.1 JAVA

Java[16] es un lenguaje de programación orientado a objetos desarrollado a principios de los años 90 por Sun Microsystems[17] (recientemente adquidira por Oracle). El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode (código intermedio más abstracto que el código máquina), aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha

controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre, aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es.

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (Common Object Request Broker Architecture), Internet Communications Engine o OSGi (Open Services Gateway Inititative) respectivamente.

El diseño de Java, su robustez, el respaldo de la industria y su fácil portabilidad han hecho de Java uno de los lenguajes con un mayor crecimiento y amplitud de uso en distintos ámbitos de la industria de la informática.

2.8.1.a En sistemas de servidor

En la parte del servidor, Java es más popular que nunca, desde la aparición de la especificación de Servlets y JSP (Java Server Pages).

Hasta entonces, las aplicaciones web dinámicas de servidor que existían se basaban fundamentalmente en componentes CGI y lenguajes interpretados. Ambos tenían diversos inconvenientes (fundamentalmente lentitud, elevada carga computacional o de memoria y propensión a errores por su interpretación dinámica).

Los servlets y las JSPs supusieron un importante avance ya que:

- El API de programación es muy sencilla, flexible y extensible.
- Los servlets no son procesos independientes (como los CGIs) y por tanto se ejecutan dentro del mismo proceso que la JVM mejorando notablemente el rendimiento y reduciendo la carga computacional y de memoria requeridas.
- Las JSPs son páginas que se compilan dinámicamente (o se pre-compilan previamente a su distribución) de modo que, el código que se consigue, supone una ventaja en el rendimiento frente a muchos lenguajes interpretados.

La especificación de Servlets y JSPs define un API de programación y los requisitos para un contenedor (servidor) dentro del cual se puedan desplegar estos componentes para formar aplicaciones web dinámicas completas. Hoy día existen multitud de contenedores libres y comerciales compatibles con estas especificaciones.

A partir de su expansión entre la comunidad de desarrolladores, estas tecnologías han dado paso a modelos de desarrollo mucho más elaborados con frameworks que se sobreponen sobre los servlets y las JSPs para conseguir un entorno de trabajo mucho más poderoso y segmentado en el que la especialización de roles sea posible tales como desarrolladores, diseñadores gráficos y se facilite la reutilización y robustez de código. A pesar de todo ello, las tecnologías que subyacen (Servlets y JSPs) son substancialmente las mismas.

Este modelo de trabajo se ha convertido en un estándar de-facto para el desarrollo de aplicaciones web dinámicas de servidor y otras tecnologías (por ejemplo. ASP) se han basado en él.

2.8.1.b Plataformas soportadas

Una versión del entorno de ejecución Java JRE (Java Runtime Environment) está disponible en la mayoría de equipos de escritorio. Sin embargo, Microsoft no lo ha incluido por defecto en sus sistemas operativos. En el caso de Apple, éste incluye una versión propia del JRE en su sistema operativo, el Mac OS. También es un producto que por defecto aparece en la mayoría de las distribuciones de GNU/Linux. Debido a incompatibilidades entre distintas versiones del JRE, muchas aplicaciones prefieren instalar su propia copia del JRE antes que confiar su suerte a la aplicación instalada por defecto. Los desarrolladores de applets de Java o bien deben insistir a los usuarios en la actualización del JRE, o bien desarrollar bajo una versión antigua de Java y verificar el correcto funcionamiento en las versiones posteriores.

2.8.2 PHP

PHP[18] es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica.

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdof en 1994; sin embargo

la implementación principal de PHP es producida ahora por “The PHP Group” y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

2.8.2.a Uso en servidores

PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin coste alguno. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de páginas web, es posible crear aplicaciones con una interfaz gráfica para el usuario. También puede ser usado desde la línea de comandos, de la misma manera como Perl o Python pueden hacerlo, a esta versión de PHP se la llama PHP CLI (Command Line Interface).

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica, por ejemplo obteniendo información de una base de datos. El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

PHP además permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX, Linux, Mac OS X y Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

Existen además herramientas alternativas de IDE (entorno de desarrollo integrado) para los

programadores, tanto comerciales como de software libre, para poder desarrollar con PHP.

2.8.3 Free Pascal y Lazarus

Free Pascal[19] es un compilador de Pascal de código abierto con dos características notables: un alto grado de compatibilidad con Delphi y disponibilidad en una variedad de plataformas, incluyendo OS X, Windows, Mac, y Linux. La compatibilidad de Free Pascal con Delphi incluye no solamente la ayuda para el mismo lenguaje de programación Object Pascal que utiliza Delphi, sino también para muchas de las mismas bibliotecas de rutinas y de clases de gran alcance por las que Delphi es conocido. Esto incluye unidades habituales tales como System, SysUtils, StrUtils, DateUtils, Classes, Variants, Math, IniFiles y Registry, que se incluyen con Free Passcal en todas las plataformas soportadas. Free Pascal también incluye unidades tales como Windows, ShellAPI, BaseUnix, Unix y DynLibs para acceder a características específicas de un sistema operativo. Este conjunto de unidades se denomina generalmente como la librería de tiempo de ejecución de Free Pascal(RTL).

Lazarus es un sistema de desarrollo de código abierto que trabaja sobre el compilador Free Pascal agregando un entorno integrado de desarrollo (IDE) que incluye un editor de código con resalte de sintaxis y un diseñador de formularios visual, así como una librería de componentes que es altamente compatible con la librería de componentes visual de Delphi (VCL). La librería de componentes de Lazarus (LCL) incluye los equivalentes para muchos de los controles familiares de VCL tales como formas, botones, cajas de texto y más que se utilizan para crear aplicaciones que tienen un interfaz gráfico de usuario (GUI).

Tanto Free Pascal como Lazarus están escritos en Pascal. El código fuente completo está disponible no solamente para el compilador de Free Pascal y el IDE de Lazarus, sino también para todas las unidades que construyan Free Pascal RTL y Lazarus LCL.

Como Delphi, Free Pascal y Lazarus son ambos herramientas de programación de uso general, lo que significa que se puede desarrollar una amplia variedad de programas con ellos, incluyendo lo siguiente:

2.8.3.a Aplicaciones de consola

Las aplicaciones de consola no tienen un GUI. En su lugar se lanzan la consola, leen su entrada y escriben generalmente su salida en la consola. En Windows la consola es la ventana del aviso de

comando. En OS X y Linux la consola es la ventana terminal. Las aplicaciones de consola incluyen cosas como utilidades pequeñas tales como el programa de Windows FC (File Compare) o los comandos de Unix cd y cp. Las aplicaciones de consola pueden también ser utilizadas por los programas de proceso de datos que no necesitan un GUI porque son arrancados por otros programas o desde archivos por lotes. El compilador Free Pascal y los programas utilitarios incluidos con él son todos aplicaciones de consola, lo que significa que pueden ejecutarse desde la consola, desde un archivo por lotes, o desde el IDE de Lazarus.

Se puede crear una aplicación de consola sin más que un editor de textos y el compilador Free Pascal. No es necesario utilizar Lazarus para desarrollar aplicaciones de consola. Sin embargo, esi preferible trabajar en un ambiente integrado, utilizando Lazarus para crear un proyecto para una aplicación de consola y editar y compilar el código en el IDE de Lazarus.

2.8.3.b Librerías cargables dinámicamente

Una Librería cargable dinámicamente es generalmente una colección de funciones compiladas que se pueden llamar por otros programas. Como el nombre sugiere, la librería no se enlaza con su ejecutable en tiempo de compilación, sino que por el contrario se carga en el tiempo de ejecución. En Windows, un archivo de la librería tiene una extensión .dll (dynamic-link library=librería de enlace dinámico, o DLL). En OS X, un archivo de librería tiene una extensión .dylib (dynamic shared library=librería compartida dinámica). En Linux, un archivo de la librería tiene una extensión .so (shared object library o librería de objetos compartidos). Las librerías cargables dinámicamente se utilizan típicamente para desarrollar complementos para otros programas, desarrollar librerías que se pueden llamar por los programas escritos en otras lenguajes tales como C y C++, o para descomponer proyectos grandes en trozos de modo que los desarrolladores del proyecto no se entorpezcan unos a otros. Windows se compone de centenares de DLLs, al igual que muchos otros programas grandes tales como OpenOffice.org.

Igual que para las aplicaciones de consola, solamente se necesita un editor de textos y el compilador Free Pascal para desarrollar una librería, aunque se puede también crear un proyecto de Lazarus para una librería y desarrollarla en el IDE de Lazarus.

2.9 Otras tecnologías y/o recursos adicionales evaluados

Además del sistema operativo y el lenguaje de programación, es necesario hacer uso de tecnologías

y servicios web que permitan dotar al MGM de los mecanismos necesarios para poder desempeñar las funciones de descarga de mapas provenientes de las solicitudes que le envía el módulo de comunicaciones del SIT, almacenamiento de mapas para poder mejorar el rendimiento ante peticiones que ya se han resuelto anteriormente y dotar de la posibilidad de acceso a los dispositivos móviles de los mapas descargados, mediante el uso de un servidor web.

Estas funcionalidades deben poder satisfacerse con las principales características que definen el desarrollo de aplicaciones cliente/servidor en red junto con servicios web, o lo que es lo mismo, priorización de rendimiento, posibilidad de escalabilidad, robustez ante fallos además de minimización de costes monetarios tanto en sistemas como en software y líneas de comunicación.

A continuación se van a detallar las alternativas de herramientas y servicios web necesarios en el sistema.

2.9.1 Socket de Internet

Socket designa un concepto abstracto por el cual dos programas situados en computadoras distintas, pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada.

Un socket queda definido por una dirección IP, un protocolo y un número de puerto.

Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de bytes, es decir, datos relevantes a su finalidad.

Para ello son necesarios los tres recursos que originan el concepto de socket:

- Un protocolo de comunicaciones, que permite el intercambio de octetos.
- Una dirección del Protocolo de Red (Dirección IP, si se utiliza el Protocolo TCP/IP), que identifica una computadora.
- Un número de puerto, que identifica a un programa dentro de una computadora.

Los sockets permiten implementar una arquitectura cliente/servidor. La comunicación ha de ser iniciada por uno de los programas que se denomina programa cliente. El segundo programa espera a que otro inicie la comunicación, por este motivo se denomina programa servidor.

Un socket es un recurso existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

Los sockets dependen de las características del protocolo en el que se implementan. El protocolo más utilizado es TCP, aunque también es posible utilizar UDP. Gracias al protocolo TCP, los sockets tienen las siguientes propiedades:

- Orientado a conexión.
- Se garantiza la transmisión de todos los bytes sin errores ni omisiones.
- Se garantiza que todo byte llegará a su destino en el mismo orden en que se ha transmitido.

Estas propiedades son muy importantes para garantizar la corrección de los programas que tratan la información.

El protocolo UDP es un protocolo no orientado a la conexión. Sólo se garantiza que si un mensaje llega, llegue bien. En ningún caso se garantiza que llegue o que lleguen todos los mensajes en el mismo orden que se mandaron. Esto lo hace adecuado para el envío de mensajes frecuentes pero no demasiado importantes, como por ejemplo, mensajes para las actualizaciones de un gráfico.

2.9.2 Servicios web para la geolocalización de datos y descarga de mapas

Para realizar la tarea de descargar un mapa correspondiente a una petición iniciada desde un dispositivo móvil, es necesario primero traducir la localización del mismo indicada por la calle, ciudad, provincia, país, código postal, a un posicionamiento dentro de un sistema de coordenadas, tratando su representación como un objeto espacial dentro del globo terráqueo. Este proceso se denomina geolocalización o georreferenciación y es utilizado frecuentemente en los SIGs (Sistemas de Información Geográfica). La geolocalización se asocia a una longitud, latitud y altura de un objeto espacial dentro del globo terráqueo.

En el presente PFC, se utiliza el término de geolocalización aunque realmente sea una pseudo-geolocalización, porque para la descarga de mapas, mediante el uso del servicio web elegido no es necesario la altura del objeto espacial, ya que, de momento el mapa está representado en un ámbito bidimensional plasmado en plano de dos dimensiones.

La creciente demanda del acceso a la información geografía en cada tipo de actividad humana

aumenta el número de servicios de geolocalización accesibles vía web. Los sistemas de información geográfica están presentes en nuestra vida cotidiana (traslado dentro de la ciudad, información meteorológica), ocio (viajes, información turística, redes sociales) o trabajo (viajes de negocios). Para ello se utilizan servicios web de geolocalización. Pero los servicios web de geolocalización se diferencian entre ellos en su nivel de detalle, cobertura, fiabilidad o precisión.

En la actualidad ante un entorno web con mucha diversidad de servicios web, uno de los problemas que se presenta, consiste en la búsqueda y evaluación de las alternativas que satisfagan en general a los usuarios, los cuales no suelen estar interesados en la procedencia de la información, pero si en su calidad.

Hoy en día existen muchos proveedores de servicios de geolocalización accesibles vía Web debido a la creciente demanda de acceso a información geográfica en contextos como la vida cotidiana (p.e. callejeros), el ocio (p.e. Redes sociales), el trabajo (p.e. portales de viajes) y los servicios públicos (p.e. sistemas de emergencia). Independiente del caso de uso se necesita información geográfica ajustada a los requisitos.

Las características básicas de cada uno de los servicios están formadas por el tipo de datos proporcionados (direcciones, puntos de interés , nombres históricos, etc), la cobertura (España, municipios o el mundo) y el nivel de detalle, llamado granularidad (ej. calle , dirección, código postal, etc.). La calidad del servicio de geolocalización está condicionada por factores dependientes de los requisitos de calidad en el servicio (QoS, Quality of Service), tiempo de respuesta, fiabilidad y de los datos (variaciones en nivel de granularidad, precisión). Además, los servicios se diferencian por el modo de acceso (libre, restringido, de pago) muy influenciado por el dominio de los servicios de geolocalización por las empresas privadas. Los servicios dedicados de pago garantizan la mejor calidad y pueden servir como soporte a los sistemas que requieren información muy precisa como los servicios de emergencia. En la oferta de los proveedores más grandes hay acceso libre con limitaciones a sus datos vía su propia página Web, generalmente en la forma de un servicio de callejero de buena calidad. Estas tácticas son aplicadas, por ejemplo, por Google. El servicio de GoogleMaps o la aplicación GoogleEarth son conocidos por la mayoría de los usuarios de la red. Otros conocidos proveedores son ViaMichelin , Yahoo y Geonames y Microsoft Map Point.

Además de un acceso vía un portal Web, estos proveedores ofrecen un acceso restringido vía servicios a sus datos aportando un API (Application Program Interface). Las restricciones suelen condicionar la visualización (la licencia exige uso de los visualizadores propios del proveedor) y

prohíben la reutilización de los datos. Además, las condiciones del uso del servicio libre perjudican la calidad de las aplicaciones construidas sobre dicho servicio, al establecer un tiempo mínimo entre una solicitud y la siguiente y un límite en el número de solicitudes por día entre otras restricciones.

El siguiente cuadro presenta las condiciones del acceso restringido de los principales proveedores de los servicios de geolocalización, en el momento de la implementación del presente PFC.

Proveedor	Control de acceso	Límite de peticiones/día	Otros límites	Restricción de uso
Google Maps	Código de acceso	1000 por IP	Tiempo mínimo entre peticiones consecutivas	Sólo en visualizador de Google Maps o utilización de javascript
Yahoo Maps	Código de acceso	50000		Visualizador de Yahoo Maps, acceso a la imagen del mapa directamiente mediante el uso de API de mapa estático o utilización de javascript
ViaMichelin	Código de Acceso	1000		Sólo en visualizador de Vía Michelin o utilización de javascript
Geoname	IP	50000		Pocas referencias en la base de datos de localizaciones
Microsoft Map Ppont	Código de acceso	Ilimitado		Aplicación de pago.

Tabla 1: Condiciones de acceso a proveedores de geolocalización

2.9.3 Almacenamiento de datos

Es indispensable dotar al sistema de las herramientas necesarias para minimizar el tiempo de respuesta de las solicitudes de geolocalización y descarga de mapas de los dispositivos móviles. En circunstancias normales ante cualquier solicitud de mapa, se debe primero geolocalizar la ubicación del lugar a visitar mediante el uso de un servicio web. Una vez obtenidos estos datos geolocalizados, es necesario obtener el mapa mediante el uso de otro servicio web.

¿Qué ocurre si a lo largo del tiempo distintos usuarios con sus dispositivos móviles solicitan el mapa de una misma localización, como por ejemplo un monumento muy conocido dentro de un itinerario turístico que realizan cientos de personas cada día?. En condiciones normales, cada vez que se solicita el mapa, habría que geolocalizar su localización (calle, avenida, plaza, ciudad, código postal, provincia, etc) y obtener el mapa a partir de la geolocalización; todo ello tantas veces como distintos dispositivos móviles lo soliciten,

Dado que la ubicación de los puntos a visitar en el SIT no cambian, se hace necesario el uso de un sistema de almacenamiento de datos, para evitar tener que realizar peticiones innecesarias, previamente calculadas en anteriores peticiones.

Para ello se implementará un sistema de cache de geolocalizaciones e imágenes de los mapas, de todos los monumentos o puntos de interés del SIT, a medida que se vayan solicitando desde los dispositivo móviles,. Esto disminuirá drásticamente el tiempo de respuesta además de reducir la carga de tráfico de red y carga del sistema en el servidor de geolocalizaciones y en el de descarga de mapas.

2.9.3.a Sistemas de cache de geolocalizaciones

Para la elección del sistema de almacenamiento de datos de caché correspondientes a las geolocalizaciones se han tenido en cuenta las tres principales bases de datos "open source" que se pueden encontrar en la actualidad.

MySQL

MySQL[20] es un gestor de base de datos sencillo de usar e increíblemente rápido. También es uno de los motores de base de datos más usados en Internet, la principal razón de esto es que es gratis para aplicaciones no comerciales, dada su licencia de software libre.

Las características principales de MySQL son:

- MySQL es un sistema fácil de instalar y configurar en servidores. Puede correr en la inmensa mayoría de sistemas operativos, por lo que junto a otro lenguaje de programación de lado de servidor de alta portabilidad como Java, PHP o Perl. Permite el desarrollo de aplicaciones web fáciles de migrar y el acceso y copia de los datos desde cualquier sistema operativo.
- Es un gestor de base de datos. Una base de datos es un conjunto de datos y un gestor de base de datos es una aplicación capaz de manejar este conjunto de datos de manera eficiente y cómoda.
- Es una base de datos relacional. Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.

- Es Open Source. El código fuente de MySQL se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL para aplicaciones no comerciales. Es por lo tanto gratuita, nos permite redistribuir una aplicación que la contenga y nos permite incluso modificar su código para mejorarla o adaptarla a necesidades requeridas.

  Además, existe la seguridad de contar con una importante cuota de mercado y de saber que es una solución estable y mantenida por un buen equipo de desarrolladores.

- Es una base de datos muy rápida, segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose, sobre todo en velocidad. Por eso es una de las bases de datos más usadas en Internet.

PostgreSQL

PostgreSQL[21] es un un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD (Berkeley Software Distribution).

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada PGDG (PostgreSQL Global Development Group).

Algunas de sus principales características son:

- Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- Instalación ilimitada: Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay coste asociado a la licencia del software. Esto tiene varias ventajas adicionales:

  - Modelos de negocios más rentables con instalaciones a gran escala.

- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costes adicionales de licenciamiento.
- Existe una importante comunidad de profesionales y entusiastas de PostgreSQL que evolucionan la tecnología de manera constante y segura.

- Estabilidad: En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad.
- Extensible: El código fuente está disponible para todos sin coste. Si se necesitase extender o personalizar PostgreSQL de alguna manera, se puede hacerlo con un mínimo esfuerzo, sin costes adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.
- Multiplataforma: PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows.
- Diseñado para ambientes de alto volumen: PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mucho mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología por las mismas razones.
- Herramientas gráficas de diseño y administración de bases de datos: Existen varias herramientas gráficas de alta calidad para administrar las bases de datos.

Como características técnicas podemos enumerar entre otras las siguientes:

- Búsquedas complejas.
- Claves de Relaciones Exteriores.
- Triggers o disparadores.
- Gestión de Vistas.
- Integridad transaccional.
- De control de concurrencia multiversión.

Firebird

Firebird[22] es un sistema de administración de base de datos relacional (o DBMS) de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en el año 2000. Su código fue reescrito de C a C++. El proyecto se desarrolla activamente.

Es multiplataforma, y actualmente puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows.

- Ejecutable pequeño, con requerimientos de hardware bajos.
- Arquitectura Cliente/Servidor sobre protocolo TCP/IP.
- Soporte de transacciones ACID y claves foráneas.
- Es medianamente escalable.
- Buena seguridad basada en usuarios/roles.
- Diferentes arquitecturas, entre ellas el Firebird incrustado (embedded server) que permite ejecutar aplicaciones monousuario en ordenadores sin instalar el software Firebird.
- Bases de datos de sólo lectura, para aplicaciones que corran desde dispositivos sin capacidad de escritura, como cd-roms.
- Existencia de controladores ODBC, OLEDB, JDBC, PHP, Perl, .Net, etc.
- Requisitos de administración bajos, siendo considerada como una base de datos libre de mantenimiento, al margen de la realización de copias de seguridad.
- Pleno soporte del estándar SQL-92, tanto de sintaxis como de tipos de datos.
- Completo lenguaje para la escritura de disparadores y procedimientos almacenados denominado PSQL.
- Capacidad de almacenar elementos BLOB (Binary Large OBjects).
- Soporte de User-Defined Functions (UDFs).
- Versión autoejecutable, sin instalación, excelente para la creación de catálogos en CD-Rom y para crear versiones de evaluación de algunas aplicaciones.

Berkeley DB

A diferencia de las bases de datos anteriormente citadas, se trata de una base de datos empotrada, que son aquellas que no inician un servicio en la máquina en la que está instalada independiente de la aplicación, pudiéndose enlazar directamente al código fuente o bien utilizarse en forma de librería.

Normalmente las bases de datos empotradas comparten una serie de características comunes: su pequeño tamaño, los pocos recursos que consumen y su rapidez a la hora de gestionar los datos.

En algunos casos las bases de datos empotradas representan el primer paso hacia un servidor de bases de datos tradicional, en otros casos simplemente su pequeño tamaño las hace ideales como sistema de soporte de información en sistemas monousuario que deba utilizar los recursos de la forma más eficiente posible, o bien como soporte a scripts de gestión de sistemas.

Berkeley DB[23] es un motor de base de datos de alto rendimiento y muy escalable que puede incluirse en cualquier aplicación. Posee un API para C, C++, Java, Perl, Python, Ruby, TCL, PHP y muchos otros lenguajes. Soporta múltiples datos para una misma clave. Berkeley DB permite miles de hilos de control manipulando bases de datos de hasta 256 terabytes en muchos sistemas, incluidos la mayoría de los tipo Unix y Windows.

Berkeley DB pertenece actualmente a Oracle[24], fue desarrollado por la compañía Sleepycat Software. Está disponible con código fuente y licencia de libre distribución. Alguna de las características del motor son:

- Los datos se almacenan en el formato nativo independiente del lenguaje de programación.
- No tiene modo cliente-servidor.
- Caché configurable para modificar el rendimiento.
- Permite crear bloqueos de forma detallada. Esto es especialmente útil para trabajos concurrentes sobre la base de datos de forma que se bloquea una página de registros durante una transacción para evitar que se modifiquen hasta que termine pero permitiendo actuar sobre el resto de páginas.
- Posibilidad de realizar copias de seguridad y replicación en caliente (sin tener que parar los servicios que la ejecutan).
- Transacciones y recuperación ante errores ACID[25]. Esto es configurable de forma que se

puede ir relajando en función de la aplicación.

- Es compatible con algunas interfaces históricas para bases de datos en UNIX como dbm, ndbm y hsearch.
- Permite utilizar la característica de snapshots para poder efectuar varias transacciones sobre los mismos registros de manera simultánea.
- Posee tres productos asociados a la marca:
  - Berkeley DB: La base de datos original escrita en C.
  - Berkeley DB Java Edition: Una versión de la anterior con algunas características menos pero con la ventaja de estar escrita en un lenguaje multiplataforma.
  - Berkeley XML DB: Edición especialmente ideada para almacenar documentos XML mediante colas XQuery. Esta versión actúa como una capa sobre Berkeley DB y tiene ramificaciones para varios lenguajes (Java, C, PHP, etc.).

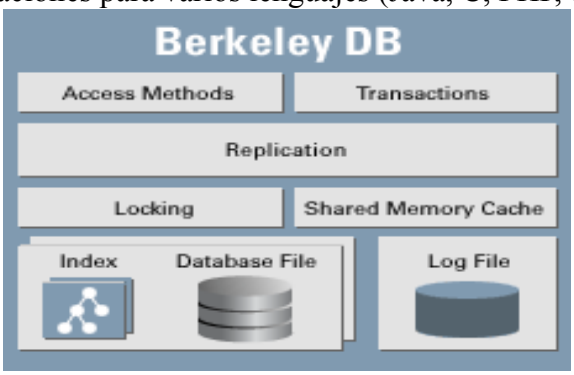

Figura 6: Esquema de la estructura de Berkeley db

Es extremadamente rápido para insertar y consultar pares de claves y valores a modo de sistema de cache de datos. Así mismo se compone de un único fichero, muy fácil de manejar a la hora de respaldarlo o cambiarlo de ubicación.

Se debe de tener en cuenta que un método de acceso se puede seleccionar sólo cuando se crea la base de datos. Una vez seleccionado, el uso del API es generalmente idéntico en todos los métodos de acceso. Es decir, la manera de utilizar las bibliotecas del API es la misma, independientemente del método de acceso elegido.

A continuación detallaremos para cada uno de los tipos de Berkeley DB, su método de acceso:

BTree	Los datos se almacenan en una estructura de árbol ordenada y equilibrada. Tanto la clave y los datos de los registros BTree puede ser arbitrariamente compleja. Es decir, que puede contener valores individuales como un entero o una cadena, o tipos complejos, como una estructura. Permite la duplicidad de registros.
Hash	Los datos se almacenan en una tabla hash lineal extendida. Al igual que BTree, la clave y los datos utilizados para el hash de los registros pueden ser arbitrariamente complejas. También, al igual que BTree, permite registros duplicados.
Cola	Los datos se almacenan en una cola como registros de longitud fija. Cada registro lógico utiliza un número de registro como su clave. Este método de acceso está diseñado para una rápida inserción al final de la cola, y tiene una operación especial que elimina y devuelve un registro del principio de la cola. Este método de acceso es inusual en el sentido de que proporciona nivel de bloqueo. Este beneficio puede proporcionar mejoras de rendimiento en aplicaciones que requieren acceso concurrente a la cola.
RecNo	Los datos se almacenan como registros con una longitud fija o registros de longitud variable. Al igual que la cola, los registros utilizan un número de registro como clave.

Tabla 2: Tipos de BerkeleyDB

2.9.3.b Sistema de cache de imágenes de mapas

No sólo es necesario el uso de sistema de cache para las geolocalizaciones de las peticiones solicitadas por parte de los dispositivos móviles canalizadas a través del módulo de comuinicaciones del SIT, sino que también se hace necesario un sistema de caché de las imágenes de los mapas descargados, para no tener que descargar desde los servidores remotos de mapas un mismo mapa correspondiente a un lugar que haya sido incluido en distintas peticiones a lo largo del tiempo.

El sistema de ficheros del sistema operativo, actuará a modo de cache de ficheros, bajo una serie de condiciones específicas que se detallan en el siguiente capítulo.

2.9.4 Servidor web, acceso a los mapas descargados

La descarga de un mapa desde un servidor remoto, implica también la utilización de alguna herramienta o tecnología para poder servir o devolver ese mapa al dispositivo móvil que originó inicialmente la petición. Para ello, la mejor solución se centra en la utilización de servidores web, que mediante el uso del protocolo HTTP puedan servir las imágenes de los mapas a los distintos dispositivos móviles.

Se han tenido en cuenta los tres principales servidores web que utilizan software libre para poder desempeñar esta tarea de la forma más eficiente.

2.9.4.a Apache

Apache[26] es el servidor web por excelencia, su alta posibilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa.

La historia de Apache se remonta a febrero de 1995, donde empieza el proyecto del grupo Apache, el cual esta basado en el servidor Apache httpd de la aplicación original de NCSA. El desarrollo de esta aplicación original se estancó por algún tiempo tras la marcha de Rob McCool por lo que varios webmaster siguieron creando sus parches para sus servidores web hasta que por medio del correo electrónico, empezaron a organizarse y se conjuntaron a la hora de implementar el mantenimiento del servidor web. Fue ahí cuando formaron el grupo Apache.

Fueron Brian Behlendorf y Cliff Skolnick quienes a través de una lista de correo coordinaron el trabajo y lograron establecer un espacio compartido de libre acceso para los desarrolladores.

Fue así como fue creciendo el grupo Apache, hasta lo que es hoy. Aquella primera versión y sus sucesivas evoluciones y mejoras alcanzaron una gran implantación como software de servidor inicialmente solo para sistemas operativos UNIX y fruto de esa evolución es la versión para Microsoft Windows .

Apache es una muestra, al igual que el sistema operativo Linux, de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar.

La licencia Apache es una descendiente de la licencias BSD, no es GPL. Esta licencia permite hacer lo que se quiera con el código fuente (incluso “forks” también denominadas ramificaciones y productos propietarios) siempre que se reconozca su trabajo.

Las principales características que le han llevado a ser tan popular en el mundo del software libre se pueden resumir en :

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver que es lo que se está instalando como servidor se puede saber, sin ningún secreto.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar

las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a éste y están disponibles para que se instalen cuando se necesiten. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un modulo para realizar una función determinada.

- Apache funciona nativamente con gran cantidad lenguajes de programación como Perl, PHP y otros lenguajes de script. PHP y Perl destacan en el mundo del script en redes. Apache interactúa con ellos tanto con soporte CGI como con soporte compilado exclusivamente para ellos. También trabaja con Java y páginas jsp.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta capacidad de configuración en la creación y gestión de logs. Permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor .

2.9.4.b Lighttpd

Lighttpd[27] es un servidor web diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Está optimizado para entornos donde la velocidad es muy importante, y por eso consume menos CPU y memoria RAM que otros servidores. Por todo lo que ofrece, Lighttpd es apropiado para cualquier servidor que tenga problemas de carga.

Lighttpd es software libre y se distribuye bajo la licencia BSD. Funciona en GNU/Linux y UNIX de forma oficial. Para Microsoft Windows actualmente hay una distribución conocida como “Lighttpd For Windows”.

Entre sus principales características podemos enumerar:

- Virtual hosting (alojar varios dominios en la misma IP).
- CGI, SCGI y FastCGI.
- Soporte para PHP, Ruby, Python y otros lenaguajes de programación.
- Entorno chroot, enjaulado para permitir mayor seguridad en el sistema.
- Cifrado SSL.

- Compresión (gzip, bzip2, …).
- Autenticación (LDAP, htpasswd, otros)
- Server Side Includes
- Consumo de memoria constante.
- Redirecciones HTTP, y reescrituras de URL.
- Puede enviar partes de un fichero (rangos) .
- Puede usar select() o poll().
- También permite otros sistema de notificación de eventos como kqueue y epoll.
- Hace estadísticas mediante RRDtool.
- Muestra un listado de ficheros cuando se entra a un directorio sin index.html.
- Redirección condicional.
- Permite módulos externos.
- Cache Meta Language.
- Acepta parte de WebDAV.

Lighttpd permite comunicarse con programas externos mediante FastCGI o SCGI, que son mejoras al CGI original (también soportado). De esta forma, se pueden usar programas en prácticamente cualquier lenguaje de programación.

Tiene una importancia especial PHP, para el que se han hecho mejoras específicas. También es habitual combinarlo con Ruby on Rails (framework o entorno de aplicaciones de código abierto escrito en el lenguaje de programación Ruby).

Según las estadísticas que se pueden encontrar “http://www.lighttpd.net/benchmark/” (y en los enlaces a comparativas no oficiales), Lighttpd es varias veces más rápido que Apache en la mayoría de pruebas.

2.9.4.c Cherooke

Cherokee[28] es un servidor web de alto rendimiento. Es muy rápido, flexible y fácil de configurar. De

hecho, es el servidor web más rápido que existe tal y como se especifica en las pruebas de rendimiento de su página web y en el siguiente punto del presente documento, correspondiente a la elección de tecnologías. Ofrece soporte para las tecnologías mas extendidas hoy en día: FastCGI, SCGI, PHP, CGI, SSI, TLS y conexiones SSL encriptadas, hosts virtuales, autenticación, sobre la codificación en tiempo real, equilibrio de carga, compatibilidad con los archivos de log de Apache, balanceador de base de datos, proxy HTTP inverso, y mucho más.

Es altamente eficiente, extremadamente ligero y proporciona una estabilidad muy elevada. Entre sus muchas características, hay una que merece especial: una interfaz de usuario amigable llamada cherokee-admin que proporciona configuración sin complicaciones de cada una de las características del servidor. Este interfaz de administración permite configurar el servidor web sin tener que preocuparse de la edición de un archivo de texto escrito con una sintaxis determinada.

A diferencia de muchos otros servidores web por ahí, Cherokee maneja muchas conexiones simultáneas, tiene una huella de memoria muy baja, e incluso puede proporcionar servicios de balanceo de carga. Es idóneo para el uso entre una amplia variedad de sistemas, desde pequeños dispositivos empotrados hasta infraestructuras de empresas grandes. También es multiplataforma, ofreciendo un rendimiento nativo para Unix, Linux y sistemas Windows.

Lo mejor de todo es que Cherokee es software libre, tiene una arquitectura bien diseñada, es totalmente modular y tiene una base de código limpia y ordenada. Cualquier usuario o desarrollador puede tener acceso al código para su estudio y modificación, lo que permite personalizarlo, modificarlo, o ampliarlo todo lo posible para satisfacer necesidades específicas. Además la comunidad de desarrolladores lo mantiene y revisa de forma constante.

3. TECNOLOGÍAS ELEGIDAS

En este punto se va a especificar cuales son las tecnologías elegidas para la implementación del MGM, de entre todas las especificadas anteriormente.

Además de tener que satisfacer las necesidades y objetivos planteados anteriormente, se deben emplear solo tecnologías que cuya licencia sea compatible con la filosofía de software libre, sin restricciones para su uso y distribución.

3.1 Sistema operativo

Todo el proyecto ha sido desarrollado y pensado para su puesta en marcha bajo un entorno GNU Linux. Concretamente ha sido desarrollado utilizando una distribución llamada Ubuntu.

Ubuntu es una distribución a su vez basada en Debian GNU Linux.

Linux es un sistema operativo tipo Unix (también conocido como GNU/Linux) que se distribuye bajo la Licencia Pública General de GNU (GNU GPL), es decir que es software libre. Es usado ampliamente en servidores y super-computadores, y cuenta con el respaldo de corporaciones como Dell, Hewlett-Packard, IBM, Novell, Oracle, Red Hat y Sun Microsystems.

Como ventajas de un sistema Linux podemos enumerar:

- Linux es básicamente un duplicado de UNIX, lo que significa que incorpora muchas de las ventajas de este importante sistema operativo.
- En Linux pueden correr varios procesos a la vez de forma ininterrumpida como un servidor de red al tiempo que un procesador de textos, una animación, copia de archivos o revisar el correo electrónico.
- Seguridad, porque es un sistema operacional diseñado con la idea de Cliente/Servidor con permisos de acceso y ejecución a cada usuario. Esto quiere decir que varios usuarios pueden utilizar una misma maquina al tiempo sin interferir en cada proceso.
- Linux es software libre, gratuito. Linux es popular entre la comunidad de programadores y desarrolladores e implica un espíritu de colaboración.
- Linux integra una implementación completa de los diferentes protocolos y estándares de red, con los que se puede conectar fácilmente a Internet y acceder a todo tipo de información disponible.
- Su filosofía y sus programas están dictados por el movimiento “Open Source" que ha venido creciendo en los últimos años y ha adquirido la suficiente fortaleza para hacer frente a los gigantes de la industria del software.
- Linux puede ser utilizado como una estación personal pero también como un potente servidor de red.
- Posee el apoyo de miles de programadores a nivel mundial.

- El paquete incluye el código fuente, lo que permite modificarlo de acuerdo a las necesidades del usuario.
- Utiliza varios formatos de archivo que son compatibles con casi todos los sistemas operacionales utilizados en la actualidad.

Cabe destacar la posibilidad de que aunque se ha elegido Linux como sistema operativo, para la implementación y puesta en marcha del MGM, éste es multiplataforma pudiendo correr en sistemas operativos como Microsft Windows o Apple Mac OsX.

3.2 Lenguaje de programación

La implementación del servidor de mapas, requiere de una tecnología cliente/servidor que permita hacer uso en la parte del servidor, de llamadas remotas a servicios web que permitan geolocalizar y obtener mapas solicitados.

Además se hace necesaria la programación del servidor de comunicaciones mediante el uso de sockets para interactuar con el módulo de comunicaciones del SIT que previamente ha recibido las solicitudes de los dispositivos móviles.

Ante estás premisas el lenguaje de programación elegido debe tener seis características esenciales: velocidad, estabilidad, seguridad, simplicidad, conectividad y operatibilidad bajo redes. A estas características, hay que añadirle un requisito acorde a la filosofía del autor, debe ser software libre. La elección final sido PHP. A continuación se detalla la verificación de cada característica anterior, añadiendo además una serie de ventajas adicionales proporcionadas por este lenguaje de programación:

- **Velocidad:** No solo la velocidad de ejecución, la cual es importante, sino además no crea demoras entre las comunicaciones entre procesos en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otro software, especialmente bajo ambientes Unix, cuando se configura como módulo de Apache o cualquier otro servidor web.
- **Estabilidad:** La velocidad no sirve de mucho si el sistema se cae cada cierta cantidad de ejecuciones. Ninguna aplicación está al cien por cien libre de errores, pero teniendo el respaldo de una increíble comunidad de programadores y usuarios se reduce drásticamente la tasa de errores del lenguaje. PHP utiliza su propio sistema de administración de recursos y

dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.

- **Seguridad:** El sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde un archivo de configuración.
- **Simplicidad:** Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.
- **Conectividad:** PHP dispone de una amplia gama de librerías, y la posibilidad de agregarle extensiones es muy fácil. Esto le permite a PHP ser utilizado en muchas áreas, tales como encriptado, gráficos, gestión de XML y otras específicas del presente PFC como son los sockets y la gestión de recursos de red.
- **Lenguaje de Redes:** PHP es el lenguaje por excelencia en el mundo de las Redes. Ampliamente utilizado en multitud de proyectos y con una implementación desde sus orígenes enfocada a poder utilizarse en entornos de red.

Como ventajas adicionales de PHP enumeramos las siguientes:

- PHP corre en (casi) cualquier plataforma utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en cerca de 25 plataformas, incluyendo diferentes versiones de Unix, Linux, Windows (95,98,NT,ME,2000,XP) y Macs. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al sistema operativo.
- La sintaxis de PHP es similar a la del C, por esto cualquiera con experiencia en lenguajes similares a C podrá entender rápidamente PHP.
- PHP es completamente expandible. Está compuesto de un sistema principal, un conjunto de módulos y una variedad de extensiones de código.
- Muchas interfaces distintas para cada tipo de servidor. PHP actualmente se puede ejecutar bajo servidores web como Apache, Cherokee, Lighttpd, IIS, etc. Otra alternativa es configurarlo como modulo CGI.
- Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL.
- Posee una gran variedad de módulos. Cuando un programador PHP necesite una interfaz

para una librería en particular, fácilmente podrá crear una API para esta. Algunas de las que ya vienen implementadas permiten manejo de gráficos, archivos PDF, Flash, bases de datos, calendarios, XML, IMAP, POP, etc.

- Rapidez. PHP generalmente es utilizado como modulo de Apache, lo que lo hace extremadamente veloz. Esta completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.
- PHP es Open Source, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además no es obligatorio pagar actualizaciones anuales para tener una versión que funcione.

Ya que se trata de un lenguaje multiplataforma, el MGM puede ser puesto en marcha en otro sistema operativo distinto a Linux,

Una vez elegido el lenguaje de programación sobre el que se realizará la implementación del módulo de gestión de mapas MGM, es necesario describir aquellas tecnologías y/o recursos adicionales candidatos a realizar los requisitos y especificaciones primarias del MGM, como son el almacenamiento de los datos y mapas, servicios de geolocalizaciones, servidor web para para servir los mapas descargados, sistemas de caché, etc.

3.3 Sistema de geolocalización y descarga de mapas

Es importante tener claro que el módulo de mapas que se implemente en el PFC debe poder obtener los mapas mediante servicios web estáticos, que no requieran el uso de lenguajes dinámicos como javascript o uso de navegadores que lo implementen, ya que, en ese caso el MGM carecería de sentido, puesto que podría implementarse directamente en el dispositivo móvil mediante el uso de navegadores web.

Además el servicio web para poder geolocalizar y descargar los mapas solicitados debe tener un API que permita usarlo directamente en el lenguaje de programación previamente elegido, debe ofrecer la descarga de mapas para scripts sin necesidad de la utilización de código javascript en un navegador web. El servicio Yahoo Maps proporciona estas necesidades básicas, por lo que finalmente ha sido elegido para desarrollar el servidor de geolocalizaciones y mapas. Además el uso del API de Yahoo Maps es muy sencillo y dispone de abundante documentación.

3.3.1 Cache de geolocalizaciones

A medida que se iba realizando el análisis y evaluando las funcionalidades requeridas a soportar dentro del MGM, la idea de implementar una caché de datos se iba haciendo necesaria.

Para la implantación de esta cache, se podía optar por utilizar un sistema gestor de base de datos relacional tipo MySql, PostgreSQL, o por el contrario un sistema de almacenamiento no sql del estilo clave/valor tipo Berkeley DB, Firebird, etc. Tras realizar unas comprobaciones y tests de rendimiento se optó por está ultima solución dado que la información a almacenar se compone de una clave que indica la dirección y su valor correspondiente a la geolocalización. Además, el rendimiento obtenido a la hora de insertar y leer los datos necesarios para la geolocalizaciones de las peticiones, es muy superior en el caso de Berkeley DB frente a PostgreSQL o MySql. También la cantidad de recursos consumidos en la operaciones de lectura y son mucho menores en la utilización de Berkeley DB.

Otra cuestión importante son las copias de seguridad. Mientras que para MySQL o PostgreSQL debe ejecutarse periódicamente un script o servicio que pueda realizar un *"dump"* o volcado o copia de los datos, en el caso de Berkeley DB sólo es necesario copiar el fichero a la ubicación deseada mediante el comando *"cp"* en GNU/Linux.

Además Berkeley DB está soportado nativamente en PHP mediante la inclusión de un módulo, proporcionando todos los mecanismos necesarios para poder operar con los datos.

Una vez que se tiene claro que se va a utilizar la tecnología de Berkeley DB para la cache de geolocalizaciones, se debe tener en cuenta sobre qué método de acceso se estructurará el fichero Berkeley DB.

Para seleccionar un método de acceso, primero se tiene que considerar lo que se desea utilizar como una clave en la base de datos. Si se desea utilizar datos como cadenas alfanuméricas la elección recae entre usar Btree o bien Hash. Si se van a utilizar números de registro lógico (esencialmente enteros), entonces se debe elegir entre la cola o recNo.

Una vez tomada esta decisión respecto al tipo de datos de la clave, queda elegir entre cualquiera de BTree o Hash, o por el contrario cola o recNo. Esta decisión se describe a continuación.

3.3.1.a Elegir entre BTree y hash

Si se trabaja con un conjunto pequeño de datos cuya totalidad puede ubicarse en la memoria ram de la máquina, no hay diferencia entre BTree y Hash. Ambos tienen un rendimiento muy similar.

Debe tenerse en cuenta, que la principal preocupación es la base de datos de trabajo, no todo el conjunto de datos. Muchas aplicaciones tienen que mantener grandes cantidades de información pero sólo es necesario acceder a alguna pequeña parte de esos datos con bastante frecuencia. Así, lo que se necesita es examinar los datos que se utilizan habitualmente, y no la suma total de todos los datos gestionados por la aplicación.

Sin embargo, si los datos crecen hasta el punto en el que no caben en memoria conjuntamente, entonces hay que tener más cuidado al elegir su método de acceso. En concreto, se debe elejir:

- BTree si las claves se pueden localizar a partir de su referencia. Es decir, si se realiza una ordenación y se espera que una consulta para una clave dada probablemente será seguida por una consulta de uno de sus vecinos.
- Hash, si el conjunto de datos es muy grande. Para cualquier método de acceso, la base de datos debe mantener una cierta cantidad de información interna. Sin embargo, la cantidad de información que se debe mantener para Hash es mucho mayor que para Btree. El resultado es que, como la base de datos crece, esta información interna puede dominar la memoria caché hasta el punto de que hay relativamente poco espacio para la aplicación de datos. Como resultado de ello,Hash puede ser obligado a realizar más operaciones de entrada y salida de disco con mucha más frecuencia frente a BTree.

  Además, ante un conjunto de datos muy grande, la base de datos debe realizar muchas operaciones de entrada y salida de disco ante peticiones al azar, entonces definitivamente Btree es la solución a adoptar, ya que tiene menos registros internos a través de la búsqueda que Hash.

3.3.1.b Elegir entre la cola y recNo

RecNo o cola se utilizan cuando la aplicación necesita hacer uso de los números de registro lógico como clave principal de la base de datos. Los números de registro lógico son esencialmente números enteros que identificar de forma única el registro de base de datos. Pueden ser fijos o variables. Si el número de registro lógico es variable, puede cambiar a medida que se insertan o eliminan registros en la base de datos. Un número fijo lógico nunca cambian, independientemente

de las operaciones que que se lleven a cabo en base de datos.

Para decidir entre la cola y recNo, se debe elegir:

- Cola si la aplicación requiere un alto grado de concurrencia. El modo de acceso por cola proporciona el bloqueo a nivel de registro (en oposición a la página que el bloqueo a nivel en la utilización de otros métodos de acceso), y esto puede dar lugar a mayor rendimiento para las aplicaciones altamente concurrentes.

  Sin embargo, sólo se proporciona apoyo para registros de longitud fija. Por lo tanto, si el tamaño de los datos que desea almacenar varía ampliamente de registro a registro, probablemente se deba elegir un método de acceso que no sea la cola.

- RecNo si desea números de registro lógico variables. El métido de acceso por cola sólo es capaz de manejar un números de registro lógico fijos. Además, recNo proporciona apoyo para las bases de datos cuyo almacenamiento es un archivo de texto plano. Esto es útil para aplicaciones que buscan rápidamente, mientras que el almacenamiento temporal de datos está siendo leído o modificado.

Finalmente evaluando las alternativas la elección del métdo de acceso ha recaído en Btree, puesto que la clave está formada por caracteres alfanuméricos, probablemente localizados a partir de su referencia y ordenados. Además todo el conjunto de datos cabe perfectamente en memoria ram y existen pocos accesos al disco.

3.3.2 Cache de imágenes de mapas

El sistema de almacenamiento y cache para las imágenes de los mapas descargados, se basa en la utilización del sistema de ficheros de Linux. Se realizará una dispersión en varios subdirectorios, teniendo en cuenta la la suma del ordinal correpondiente a cada uno de los caracteres ascii de la latitud, longitud, y zoom del mapa para luego aplicarle la division entre 256 y finalmente seleccionar el valor resto de la división, con lo cual somo máximo existirán 256 subdirectorios en los que se recogeran todas las imágenes de los mapas descargados.

Con la dispersión aplicada se garantiza la eficiencia del sistema de archivos, al no estar todos los mapas dentro de un solo directorio, manteneniendo una tabla de inodos relativamente pequeña. Además se minimizan los tiempos de acceso a la los archivos de las imágenes, ya que, aplicando la función hash de dispersión se obtiene de manera muy rápida el subdirectorio en el que se encuentra

la imagen.

La nomeclatura del fichero correspondiente al mapa, está estructurada de la siguiente manera: zoom_latitud_longitud_altoimagen_anchoimagen.extensión. La extensión podrá ser gif o png.

3.4 Cómo servir las imágenes de los mapas a los dispositivos móviles

Una vez que se haya geolocalizado la ubicación proveniente por el módulo de comunicaciones del SIT, se haya descargado el mapa en repositorio local de la máquina que corra el MGM, y se haya notificado la ubicación de la imagen otra vez al módulo de comunicaciones para que éste a su vezz la retransmita al dispositivo móvil, existen varias alternativas y protocolos para que el dispositivo móvil sea capaz de obtener la imagen:

1. Mediante el protocolo FTP (**F**ile **T**ransfer **P**rotocol) o protocolo de transmisión de ficheros, en el que el dispositivo móvil se conecte al servidor FTP y adquiera el mapa correspondiente.
2. Mediante un servidor web que proporcione la imagen solicitada por el dispositivo móvil a través del protocolo HTTP (**H**iper**t**ext **T**ransfer **P**rotocol) o protocolo de transferencia de hipertexto. Esta solución requiere menos complejidad en el lado del dispositivo móvil, además de ser un protocolo con tolerancia a fallos al ir bajo TCP/IP como se ha detallado en el aparatado 2.3.2

De esta forma el servidor de comunicaciones del MGM para cualquier petición del módulo de comunicaciones del SIT, envía al mismo la respuesta en formato XML con la URL de la ubicación de la imagen del mapa para su descarga, mediante el protocolo seleccionado.

Es mucho más eficaz y fácil la utilización de un servidor web para que los dispositivos móviles

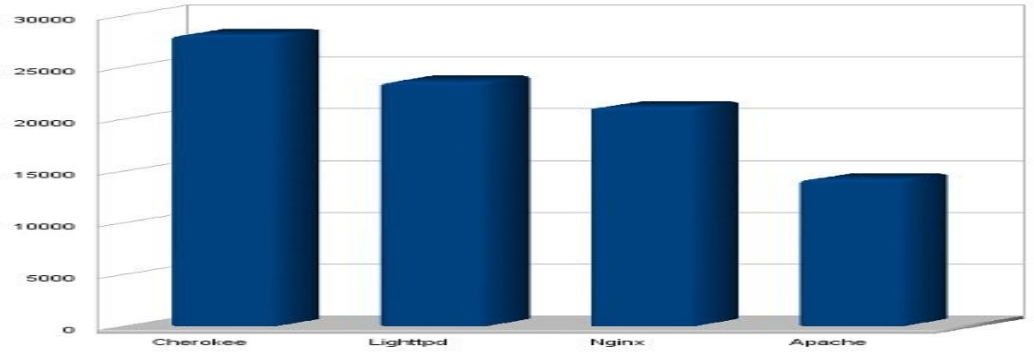

Figura 7: Rendimiento de servidores web

puedan acceder a las imágenes. Existen multitud de servidores web disponibles en la actualidad. Ante todo el servidor web elegido debe cumplir una serie de premisas:

- Debe poder ejecutarse bajo GNU/Linux
- Debe ser software libre.
- Debe ser un servidor muy ligero, que ocupe pocos recursos en la máquina, dado que sólo va a servir contenido estático, en concreto las imágenes de los mapas.
- Debe poder ser escalable, de tal forma que ante un futuro en el que haya demasiados clientes (dispositivos móviles) el servidor sea capaz de dar servicio a todos ellos.
- Debe ser fácilmente configurable.
- Opcionalmente ante ante posibles necesidades futuras, debe poder ejecutar otro tipo de servicios tales como la ejecución de contenido dinámico.

Las posibles alternativas de servidores web descritas anteriormente son Cherokee Web Server, Apache Web Server y Ligthttpd Web Server.

El rendimiento para servir contenido estático entre Cherokee y Ligthttpd es muy similar, siendo algo menor que en Apache. Además los recursos consumidos por la máquina corriendo Apache son mucho mayores que las otras dos alternativas.

Puesto que Cherokee proporciona un servicio de configuración visual muy fácil de utilizar a través del navegador, esto hizo que finalmente fuese el servidor web elegido para proporcionar los mapas a los distintos dispositivos móviles que se conecten.

Como comparativa se muestra una imagen en las que podemos ver claramente el rendimiento de estos tres servidores web.

3.5 Libre Office

LibreOffice es una suite ofimática de software libre y código abierto de distribución gratuita que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos. Está disponible para muchas y diversas plataformas, como Microsoft Windows, sistemas de tipo Unix con el Sistema X Window como GNU/Linux, BSD, Solaris y Mac OS X. LibreOffice está pensado para ser compatible con Microsoft Office, con quien

compite. Soporta el estándar ISO OpenDocument con lo que es fácil el intercambio de documentos con muchos programas, y puede ser utilizado sin costo alguno.

Se ha utilizado concretamente LibreOffice Writer como procesador de textos para la redacción del presente documento.

3.6 GIMP

GIMP (GNU Image Manipulation Program) es un programa de edición de imágenes libre y gratuito, englobado en el proyecto GNU y disponible bajo la licencia Licencia pública general de GNU.

La primera versión de GIMP se desarrolló para sistemas Unix y fue pensada especialmente para GNU/Linux. Existen versiones totalmente funcionales para Windows, para Mac OS X y para otros sistemas operativos. Se le puede considerar como la alternativa más firme para Photoshop, aunque posee una interfaz muy diferente.

3.7 VI

vi es un simple editor de texto, que no lo formatea en absoluto, pues no centra ni justifica párrafos pero permite mover, copiar, eliminar o insertar caracteres por medio del búffer permaneciendo la información ahí hasta que los cambios en el archivo se hayan guardado o bien hasta que termine la ejecución de la aplicación sin haber guardado las modificaciones.

vi fue originalmente escrito por Bill Joy en 1976, tomando recursos de ed y ex, dos editores de texto deficientes para Unix, que trataban de crear y editar archivos, de ahí, la creación de *vi*.

Hay una versión mejorada que se llama *vim*, pero *vi* es un editor de texto que se encuentra en (casi) todo sistema de tipo Unix, de forma que conocer rudimentos de v*i* es una salvaguarda ante operaciones de emergencia en diversos sistemas operativos.

3.8 Yed Graph Editor

Es un software multiplataforma, basado en Java que simplifica la realización de diagramas y esquemas de figuras y gráficos, utilizadas en el presente proyecto final de carrera. Hay muchas otras soluciones similares en el mercado, pero el tipo de licencia de utilización de la herramienta,

perfectamente bajo la la licencia de software libre, ha permitido que el autor del proyecto se decante por esta herramienta de diseño.

Puede obtenerse más información en http://www.yworks.com/en/products_yed_about.html.

3.9 Doxygen

Doxygen es un generador de documentación para C++, C, Java, Objective-C, Python, y en cierta medida para PHP, C# y D. Dado que es fácilmente adaptable, funciona en la mayoría de sistemas Unix así como en Windows y Mac OS X.

Permite generar en distintos formatos (Latext, html, rtf, etc) la documentación necesaria desde múltiples directorios con los ficheros que contienen código fuente.

La salida con los ficheros correspondientes a la documentación está formateada por una serie de plantillas totalmente editables por el usaurio.

En el presente PFC la documentación generada recoge tanto el código fuente, como desccripción y gráficos de las clases utilizadas y múltiples imágenes con los flujos de utilización y llamadas de las diferentes funciones.

4. VALORACIÓN DEL PROYECTO

En este punto se va a detallar la valoración para la implementación y puesta en marcha del MGM, teniendo en cuenta los recursos humanos, máquinas y software empleado, obteniendo finalmente el coste económico del proyecto.

La siguientes figura detalla la planificación en tiempo de las distintas fases de ejecución del proyecto:

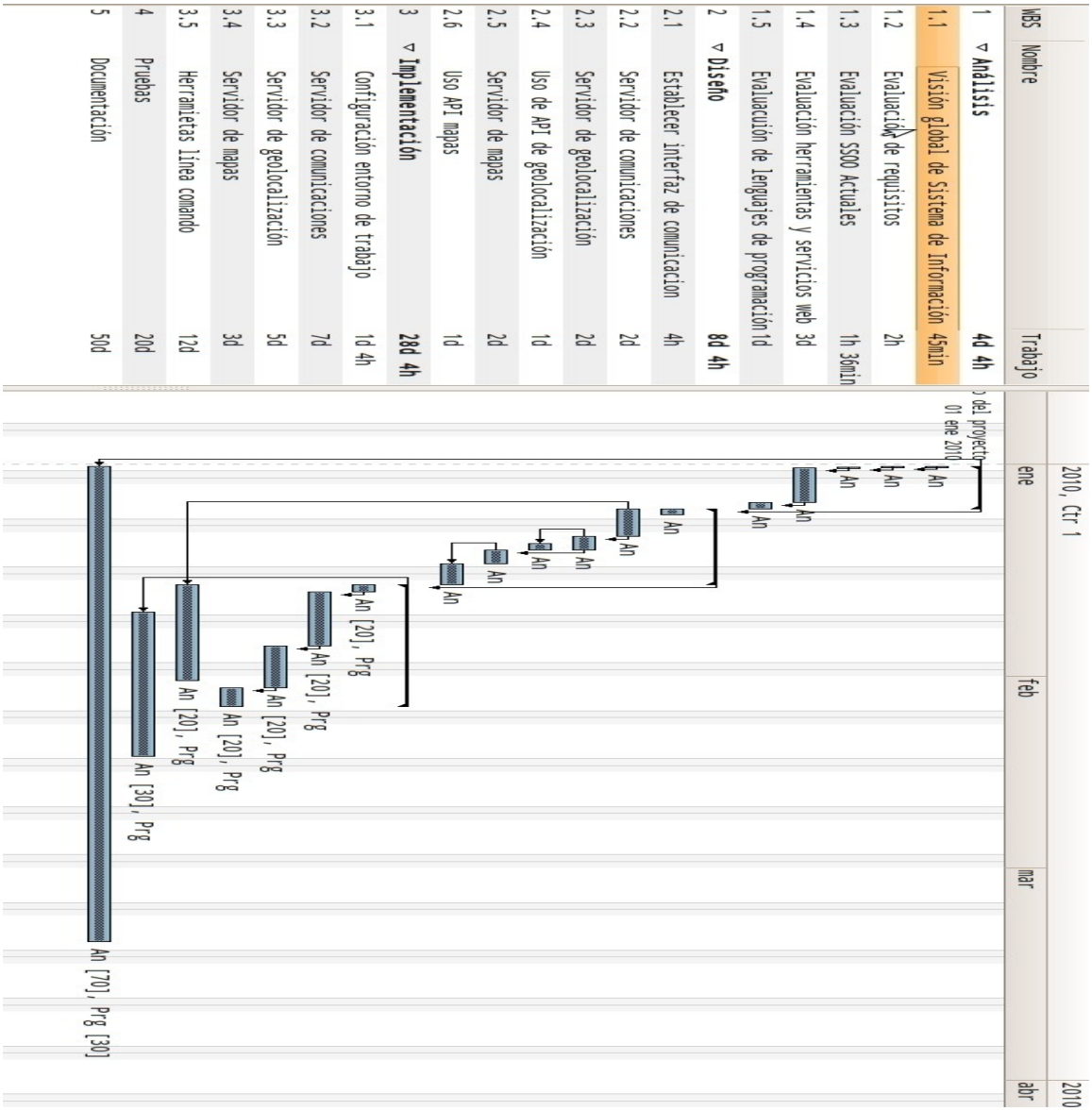

Figura 8: Diagrama Gantt. Fases del MGM

Como recursos humanos, se establecen las figuras de analista y programador, asignando un porcentaje de cada uno de estos roles a las tareas de toma de requisitos y balance de situación, análisis, implementación, pruebas y puesta en marcha del MGM.

A continuación se especifican los recursos humanos y sus costes económicos por hora en euros.

Nombre	Nombre (	Tipo	Grupo	Correo-e	Coste
Analista	An	Trabajo			28
Programador	Prg	Trabajo			15

Figura 9: Coste monetario por hora de los recursos humanos

Teniendo en cuenta los costes de los recursos y las fases del proyecto se detalla una tabla con las valoración por tareas y recursos implicados en ella.

WBS	Nombre	Inicio	Fin	Trabajo	Duració	Desperd	Coste	Asignado a	% Comple
1	**Análisis**	**ene 1**	**ene 7**	**4d 4h**	**4d 4h**		**1018,1**		100
1.1	Visión global de Sistema de Información Turística	ene 1	ene 1	45min	45min		21	An	100
1.2	Evaluación de requisitos	ene 1	ene 1	2h	2h		56	An	100
1.3	Evaluación SSOO Actuales	ene 1	ene 1	1h 36mi	1h 36mi		45,11	An	100
1.4	Evaluación herramientas y servicios web	ene 1	ene 6	3d	3d		672	An	100
1.5	Evaluacuión de lenguajes de programación	ene 6	ene 7	1d	1d		224	An	100
2	**Diseño**	**ene 7**	**ene 18**	**8d 4h**	**7d**	**7d 3h**	**1904**		100
2.1	Establecer interfaz de comunicacion	ene 7	ene 8	4h	4h	13d 7h	112	An	100
2.2	Servidor de comunicaciones	ene 7	ene 11	2d	2d	5d 3h	448	An	100
2.3	Servidor de geolocalización	ene 11	ene 13	2d	2d	5d 3h	448	An	100
2.4	Uso de API de geolocalización	ene 12	ene 13	1d	1d	10d 3h	224	An	100
2.5	Servidor de mapas	ene 13	ene 15	2d	2d	5d 3h	448	An	100
2.6	Uso API mapas	ene 15	ene 18	1d	1d	7d 3h	224	An	100
3	**Implementación**	**ene 18**	**feb 5**	**28d 4h**	**13d 6h**	**5d 5h**	**3914**		100
3.1	Configuración entorno de trabajo	ene 18	ene 19	1d 4h	1d 2h	7d 1h	206	An, Prg	100
3.2	Servidor de comunicaciones	ene 19	ene 27	7d	5d 6h	7d 3h	961,33	An, Prg	100
3.3	Servidor de geolocalización	ene 27	feb 2	5d	4d 1h	7d 1h	686,67	An, Prg	100
3.4	Servidor de mapas	feb 2	feb 5	3d	2d 4h	5d 5h	412	An, Prg	100
3.5	Herramietas línea comando	ene 18	feb 1	12d	10d	9d 3h	1648	An, Prg	100
4	Pruebas	ene 22	feb 12	20d	15d 3h	19d	2880,15	An, Prg	100
5	Documentación	ene 1	mar 11	50d	50d		9640	An, Prg	100

Figura 10: Tabla de fases y costes

Finalmente, la siguiente figura, resume la participación de cada uno de los recursos en cada una de las tareas que conforman las realización del MGM.

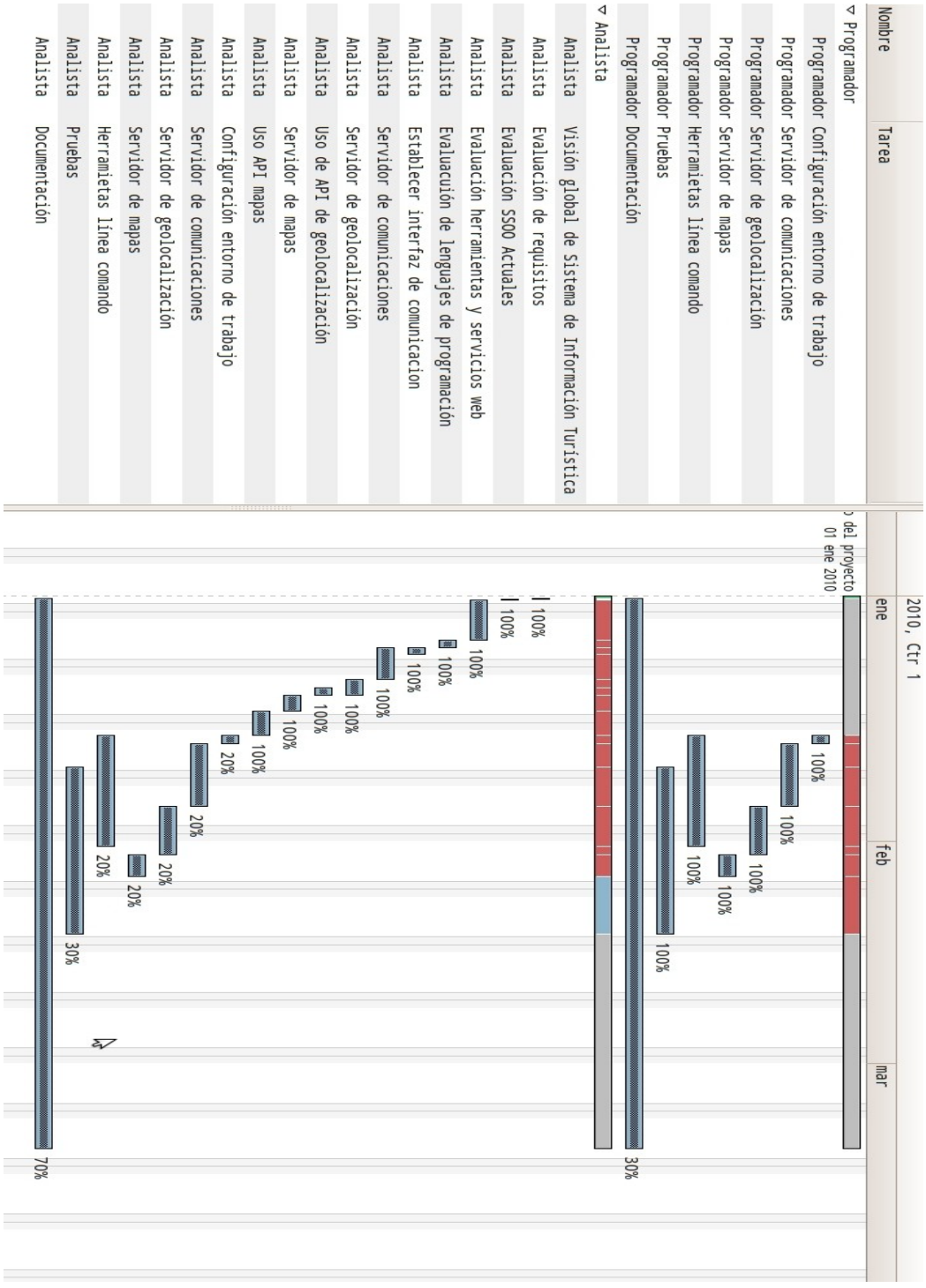

Figura 11: Uso de recursos por fase

Respecto al hardware, las máquinas elegidas, no deben tener unas especificaciones muy elevadas, ya que el sistema operativo, el lenguaje de programación y las tecnologías elegidas como la cache

de geolocalizaciones y mapas, permiten tener todo el contenido ya pregenerado, facilitando rendimiento y rapidez a la hora de realizar los objetivos especificados.

Puesto que los requerimientos de procesador y memoria no son muy elevados bastaría con una sola máquina con 2Gb de memoria RAM y un microprocesador de doble núcleo que supere los 2 Ghz. En cuanto a la elección del sistema de almacenamiento, sería recomendable tener 250 Gb de disco duro a una velocidad de 7200 rpm. Estos requisitos se corresponden con un ordenador de escritorio realizando trabajos en modo servidor, cuyo coste medio situamos en unos 500€.

Los costes de adquisición de licencias y uso son inexistentes debido al uso de herramientas y tecnologías basadas en el software libre. De hecho el coste del sistema operativo elegido y las herramientas necesarias para establecer un entorno del funcionamiento del MGM es de 0€.

Como resumen final podemos establecer un coste económico total de 19.856 €, para la implementación y puesta en marcha del MGM, calculado en base a los costes por recursos humanos (19.356 €) y máquinas (500 €) y software y herramientas utilizadas (0€).

5. GESTIÓN DEL PROYECTO

En el presente capítulo del PFC, se detalla todo el trabajo llevado a cabo para el correcto desarrollo del MGM.

En el primer punto del capítulo se expone las consideraciones previas antes de la implementación del MGM. En el segundo punto se aborda la arquitectura del MGM a desarrollar, utilizando distintos gráficos en los que se mostrará la arquitectura diseñada.

5.1 Consideraciones previas

Hoy en día la programación en entornos de red requiere una serie de conocimientos por parte del programador totalmente distintos a la programación tradicional en modo cliente/servidor.

Aunque la programación de redes bajo Internet realmente se basa en un entrono cliente/servidor mediante comunicación distribuida o directa entre el cliente y el servidor, en la parte del servidor se debe intentar cumplir una serie de requisitos que permitirán que el servicio web implementado sea robusto, cumpla sus objetivos, y pueda crecer ante la necesidad de funcionalidades futuras.

Es de vital importancia elegir el protocolo de comunicación a utilizar entre el cliente y el servidor. En el MGM se ha optado por implementar el servidor de comunicaciones utilizando el protocolo TCP/IP por ser un protocolo orientado a conexión, fiable y robusto ampliamente utilizado en entornos de red.

El servidor debe realizar los objetivos para los que ha sido implementado de la forma más eficientemente posible, para que no existan tiempos de respuesta excesivos. Para ello, se ha optado por lanzar distintos procesos hijos por cada petición para que los clientes no tengan que esperar a obtener la respuesta a su petición hasta que las peticiones anteriores hayan sido procesadas. Además, se ha optado por implementar un sistema de cache a la hora de realizar las resoluciones de las geolocalizaciones y la obtención de los mapas.

La utilización del lenguaje de programación PHP permite implementar un servicio web con un lenguaje de fácil aprendizaje, orientado al entorno de redes, ampliamente utilizado por la comunidad de desarrolladores y fácil de configurar.

Se debe utilizar en la medida de lo posible un estándar para el intercambio de información estructurada entre diferentes plataformas en el entorno de red. De ahí que se haya optado por la utilización de XML (siglas en inglés de Extensible Markup Language o "lenguaje de marcas") para la comunicación entre el MGM y el módulo de comunicaciones del SIT.

No debe olvidarse que el sistema operativo juega un papel fundamental en la programación de aplicaciones. Si se trata de redes y comunicaciones distribuidas, entonces debe tenerse muy en cuenta el sistema operativo Unix.

Linux ofrece todas las necesidades técnicas necesarias para la programación de servicios web como el que se describe en el presente PFC, utilizando herramientas estándar como procesadores de texto, un entorno de programación, la pila de protocolos de comunicaciones y la posibilidad de poder instalar nuevos paquetes o aplicaciones según las necesidades del programador, además de ser de un sistema muy estable con una comunidad de desarrolladores y empresas que contribuyen a la realización y uso global del mismo.

Otro punto a tener en cuenta es la elección del servicio web proveedor para realizar las tareas de geolocalización y descarga de mapas. Es muy importante que el servicio web que resuelva estas necesidades sea lo suficientemente rápido, estable, preciso y escalable para poder englobarlo dentro de MGM.

El MGM debe ser configurable dependiendo de ciertos parámetros especificados en ficheros de configuración, con el objetivo de poder cambiar el comportamiento del programa sin recompilar ni volver a ejecutar el código fuente.

Por último y no menos importante todos los puntos anteriores deben ser software libre, como parte de la filosofía seguida por el autor de presente PFC.

5.2 Análisis del proyecto

Para realizar el desarrollo del MGM como parte o módulo dentro de un SIT, se ha tenido que desglosar sus funcionalidades en procesos básicos que deben llevarse acabo para la consecución de los objetivos finales detallados en el punto 1.2 Objetivos.

Cada una de estas funcionalidades básicas, se encapsula en la implementación de un submódulo llamado servidor. Existen pues, tres servidores que conforman las estructura del MGM, el servidor

de comunicaciones, el servidor de geolocalilzación y el servidor de mapas.

También existe un servidor web que se encarga de servir las imágenes de los mapas descargados por el servidor de mapas.

Además, para que tanto el servidor de geolocalizaciones como el servidor de mapas eviten tener que realizar procesos que ya se han realizado anteriormente para una misma solicitud de datos, se optado por la implementación de un sistema de cache en cada uno de ellos, mejorando sustancialmente el rendimiento global del MGM y minimizando los tiempos de respuesta de las solicitudes ejecutadas desde el módulo de comunicaciones del SIT, a su vez originadas por los dispositivos móviles .

5.2.1 Servidor de comunicaciones

El servidor de comunicaciones es el módulo que gestionará todas las comunicaciones con el módulo de comunicaciones del SIT, mediante el uso de sockets utilizando el protocolo TCP/IP. Debe ser un servidor tolerante a fallos y muy rápido para minimizar los tiempos de respuesta en las peticiones. Además es el encargado de interactuar con los otros dos servidores (geolocalización y mapas) y enviar la respuesta a los dispositivos móviles a través de los sockets establecidos.

Para dotar a este servidor de múltiples opciones y funcionalidades en base a posibles situaciones que se puedan dar en el mundo real, se puede configurar de tal manera que pueda actuar en modo secuencial o paralelo, atendiendo las solicitudes de una en una, o por el contrario, de forma concurrente atendiendo varias a la vez. Adicionalmente se puede configurar la dirección IP y puerto de escucha de los sockets por los que se comunica el servidor.

Previamente se ha establecido una interfaz de datos para las comunicaciones entre los distintos dispositivos móviles el módulo de comunicaciones del SIT y el MGM, reflejada en un flujo de datos en formato XML tanto para la solicitud como para la respuesta a la solicitud. Más adelante, una vez que se aborde el servidor de geolocalizaciones, se podrá ver la estructura del flujo de datos XML.

5.2.2 Servidor de geolocalización

El servidor de geolocalización, es el encargado de parsear el flujo de datos con formato XML proporcionado por el servidor de comunicaciones, obtener la geolocalización y devolverla al

servidor de comunicaciones. En el proceso de geolocalización se realizan peticiones a un servicio web remoto que devuelve la geolocalización de las mismas.

Se ha implementado el uso de memoria cache para que solicitudes que ya se han realizado anteriormente no tengan que volver a obtenerse desde el servicio web de geolocalización remoto. Esto reduce considerablemente los tiempos de respuesta de las solicitudes.

Al igual que el servidor de comunicaciones, posee varias opciones para su configuración, tales como el tiempo máximo de permanencia en cache de las solicitudes, la ruta de almacenamiento del fichero Berkeley DB que se utilizará a modo de cache, el usuario y clave del servicio de geolocalización remoto, etc.

5.2.3 Servidor de mapas

El servidor de mapas, se encarga de obtener las imágenes de los mapas correspondientes a las geolocalizaciones obtenidas por el servidor de geolocalizaciones, y devolver al servidor de comunicaciones la ruta de almacenamiento de las mismas. Los mapas se obtienen también desde un servicio web remoto, se descargan y se almacenan en un repositorio específico, correspondiente a un directorio del sistema de ficheros de la máquina, calculado mediante una función de dispersión.

También se ha implementado un sistema de caches para no tener que volver a obtener desde el servicio web remoto mapas ya tratados anteriormente. Tiene múltiples opciones de configuración, tales como el tiempo de vigencia de los ficheros en la cache, el zoom, la altura, anchura y extensión por defecto de las imágenes a descargar, el usuario y clave del servicio de descarga de mapas remoto, etc.

5.2.4 Servidor web

Por último, existe un servidor web que se encarga de servir las imágenes de los mapas descargados por el servidor de mapas una vez, que el servidor de comunicaciones ha devuelto al módulo de comunicaciones del SIT la ubicación del mapa, y éste a su vez a los dispositivos móviles. Cuando del dispositivo móvil recibe la ubicación del mapa, la solicita al servidor web.

A continuación se muestra una figura con el flujo de datos entre los servidores del MGM, en la que se puede observar el flujo de información, desde que se realiza una petición hasta que se obtiene sus resultados.

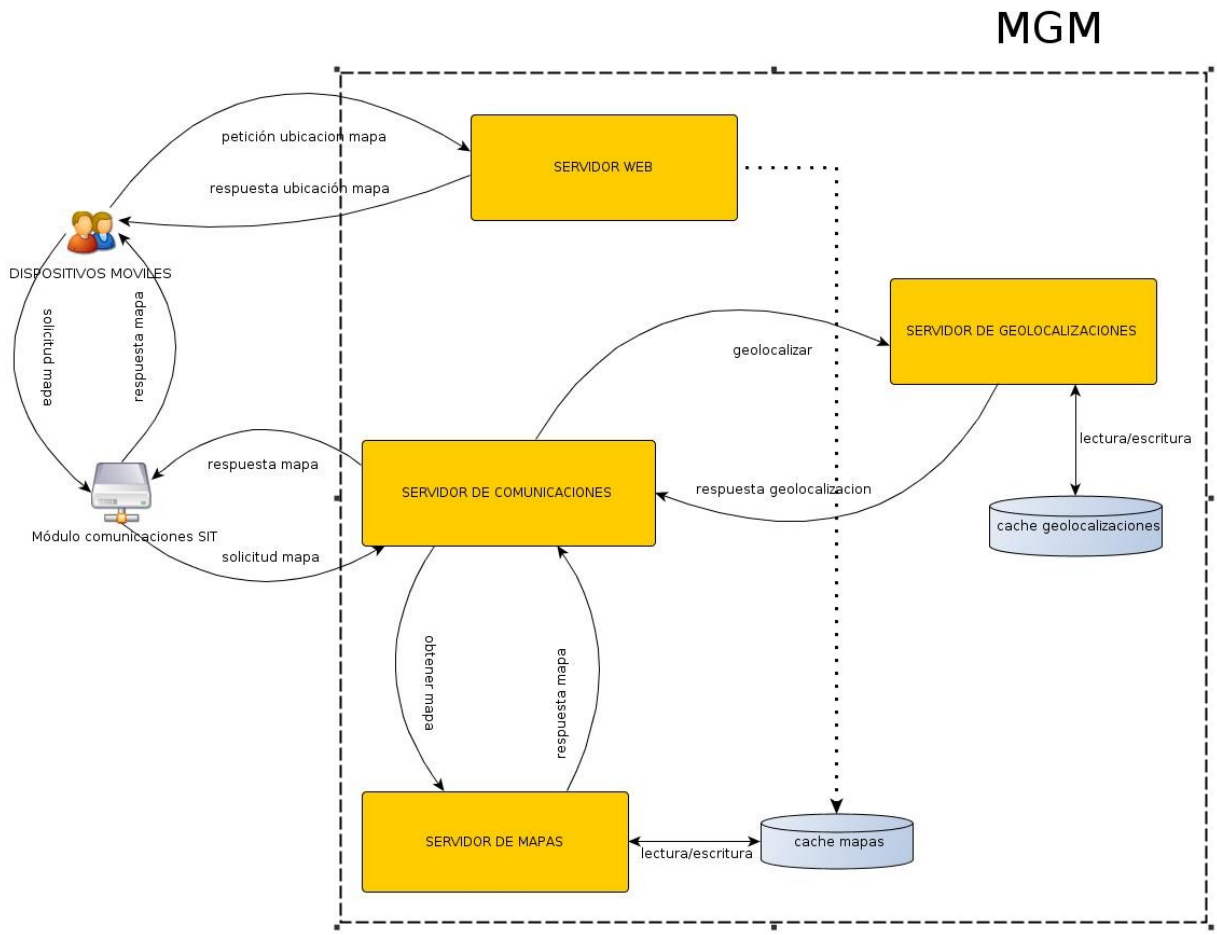

Figura 12: Diagrama global del MGM

5.3 Implementación

El presente punto se detallará la arquitectura de cada uno de los servidores, así como las ubicaciones de los archivos y directorios que lo conforman.

5.3.1.a Estructura de directorios

Cada uno de los distintos ficheros que conforman el MGM, se encuentra en una serie de directorios y subdirectorios perfectamente estructurados, almacenando las librerías de código fuente, datos de configuración, mapas descargados, cache de geolocalizaciones, fichero de logs, etc. A continuación se especifica el significado de dichos directorios.

1. Directorio base: es el directorio principal que contiene los subdirectorios en los que se engloba todo el MGM.

2. Directorio base/lib/php/MGM: es el directorio en el que se guarda todo el código fuente en formato PHP implementado en el MGM.
3. Directorio base/datos/MGM: es el directorio de datos del MGM. Este directorio contiene una serie de subdirectorios que contienen entre otros los ficheros de configuración de los distintos servidores del MGM, los logs del sistema, las geolocalizaciones y las plantillas de los mensajes de respuesta por parte del servidor de comunicaciones.
4. Directorio base/datos/MGM/conf: directorio que contiene los ficheros de configuración del servidor de comunicaciones ("fichero_configuracion_servidor.ini"), servidor de geolocalizaciones ("fichero_configuracion_geolocalizador.ini") y servidor de mapas ("fichero_configuracion_servidor_mapa.ini").
5. Directorio base/datos/MGM /geolocalizaciones: directorio en el que se encuentra el fichero de base de datos Berkeley DB que contiene la cache las distintas geolocalizaciones del MGM. El nombre del fichero es "cache_geolocalizaciones.db" aunque se puede configurar.
6. Directorio base/datos/MGM/logs: se corresponde con el directorio que almacena los ficheros log del MGM, en concreto uno por cada servidor.
7. Directorio base/datos/MGM/templates: directorio en el que se encuentran la plantilla de ("respuesta_servidor_mapas.tpl"), que servirán de respuesta a las distintas peticiones solicitadas al MGM por parte de los dispositivos móviles.

5.3.1.b Servidor de comunicaciones

Este servidor recibe desde el módulo de comunicaciones del SIT las distintas peticiones de los dispositivos móviles mediante el uso de sockets. Sirve de enlace entre las peticiones y los otros servidores del MGM. Además es el encargado de devolver al módulo de comunicaciones del SIT, las correspondientes respuestas a cada una de las peticiones solicitadas.

Las peticiones solicitadas por los dispositivos móviles, se corresponden con ubicaciones de lugares a geolocalizar por parte del SIT (monumentos, puntos de interés, etc). El servidor de comunicaciones escucha todas las conexiones mediante un socket en una dirección IP y un puerto y las transmite al módulo de geolocalización para obtener geolocalización de la misma, a continuación transmite esa geolocalización al servidor de mapas, para que éste proceda a su descarga. Una vez descargado el mapa, el servidor de comunicaciones devuleve la ubicación del

mapa al módulo de comunicaciones del SIT, para que éste a su vez lo transmita al dispositivo móvil en que originó la petición.

Las solicitudes recibidas por el servidor de comunicaciones son texto plano, con la definición de un fichero XML con codificacion UTF-8, que se detalla más adelante en el servidor de geolocalizaciones.

Como ya se ha especificado en el apartado anterior 5.1 Consideraciones previas, el protocolo utilizado para las comunicaciones entre el SIT y el MGM es TCP/IP, aportando una comunicación fiable, libre de errores, sin pérdidas de información y con seguridad frente a agentes externos.

El servidor de comunicaciones hace uso de sockets, para el intercambio de flujo de datos sobre redes TCP/IP. Un socket está asignado a una dirección IP, un puerto y un protocolo, en este caso TCP/IP.

Se puede dar el caso de necesitar varios servidores de comunicación corriendo en distintas direcciones IP y puertos. Si esto ocurre se puede utilizar el mismo script de PHP con el uso de ficheros de configuración, en los que se especifican distintas opciones del servidor a ejecutar (dirección IP, puerto de escucha, número máximo de clientes a atender, etc).

El primer paso que ejecuta el servidor de comunicaciones, es la creación de un socket por el que siempre se escucharán las peticiones de los usuarios y se dará respuesta a ellas. Una vez creado el socket se pone el servidor en “modo escucha”, quedándose dormido hasta que haya un flujo datos que atender en el socket. Existen distintas políticas a la hora de atender las peticiones en el uso de sockets:

- Atención de cada solicitud de forma secuencial, almacenando cada solicitud en una cola de peticiones. Se atiende las peticiones una a una, según el orden de llegada y no se tratará la siguiente petición hasta que no se haya procesado la anterior.
- Atención de cada solicitud de forma paralela, lanzando procesos hijos, en los que cada petición recibida se atiende inmediatamente, sin depender de la resolución de peticiones anteriores. En sistemas UNIX, lanzar un proceso hijo es algo relativamente sencillo. El crear un proceso hijo supone hacer una copia del programa y los registros de memoria del proceso padre, por lo que es bastante recomendable establecer un límite máximo de hijos a crear. Cada proceso hijo al poseer su memoria independiente, podrá seguir ejecutar las acciones necesarias para resolver la petición, devolver el resultado por el socket y morirse pasando el

control al padre. Como ventaja respecto a la política anterior, para resolver un determinado número de peticiones se emplea menos tiempo, pero por el contrario el número de recursos aumenta al lanzar procesos hijos.

El servidor de comunicaciones puede funcionar con cualquiera de estas políticas, siendo recomendable utilizar la segunda política de atención de las peticiones, ya que, en previsión de un gran número de peticiones simultáneas por parte de distintos dispositivos móviles, el MGM deber ser lo más rápido posible, minimizando los tiempos de respuesta.

Una vez que se ha recolectado los datos (flujo de datos con formato *XML*) de una petición por parte de un proceso hijo, se pasarán estos datos de la petición al servidor de geolocalizaciones, para que éste a su vez le devuelva la petición geolocalizada o un error en caso de no haber podido geolocalizarla.

Una vez que el servidor de comunicaciones tenga los datos geolocalizados, se los pasará al servidor de mapas para que éste último proceda a la obtención y descarga del mapa correspondiente a la geolocalización. El servidor de mapas devolverá al servidor de comunicaciones la ruta en la que se encuentra el mapa descargado o un error en caso de no poder asociar la geolocalización al mapa.

Cuando el servidor de comunicaciones conozca la ruta del mapa descargado, la devolverá por el socket al servidor de comunicaciones del SIT y éste al cliente del dispositivo móvil que inicialmente solicitó la petición, para que pueda acceder al mapa mediante el uso del servidor web desde cualquier localización remota.

En el caso de que el servidor de comunicaciones haya obtenido cualquier error por parte del servidor de geolocalizaciones o el servidor de mapas, también lo devolverá como respuesta a la petición. El flujo de datos que conforma la respuesta, es en realidad texto plano, con la especificación de una entidad *XML*, cuyo detalle se especifica a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
    <respuesta>
        <hay_error>0</hay_error>
        <info>http://direccion_servidor/ruta_imagen_mapa</info>
    </respuesta>
```

A continuación se detalla el significado de cada una de las líneas:

- <?xml version="1.0" encoding="UTF-8"?>
  Etiqueta que indica el inicio del fichero XML, especificando su versión y su tipo de

codificación. Son datos necesarios para la inicialización del parser del xml que trate la respuesta en el SIT. Es necesario que el flujo de datos se encuentre en codificación UTF-8.

- <respuesta>
  Etiqueta de inicio de los datos de la respuesta.
- <hay_error>0</hay_error>
  Etiqueta que indica que se ha producido cualquier tipo de error. Normalmente son errores producidos por no poder geolocalizar la petición, o la imposibilidad de calcular u obtener el mapa de la ubicación solicitada.

  Los posibles valores de la etiqueta son:

  - 0 (cero), el caso de no haber errores.
  - 1 (uno) en caso de haberse producido un error.
- *<info>http://url_imagen_mapa</info>*
  Etiqueta que contiene la respuesta a la petición. Los posibles valores son:

  - Si la etiqueta *<hay_error>* vale "0", una URL con la localización del mapa de la petición geolocalizada. El mapa contenido en la URL será servido por el servidor web.
  - Si la etiqueta *<hay_error>* vale "1", el detalle del error, especificando si no se ha podido geolocalizar, no se ha podido descargar el mapa o la petición formulada es incorrecta.

Existe la posibilidad de configurar y particularizar ciertas características del servidor de comunicaciones a la hora de ejecutarse. Dichas particularizaciones estarán contenidas en un fichero de configuración (fichero de texto con un formato determinado), incluido en el directorio de ficheros de configuración del MGM. El formato del fichero es el siguiente:

[DATOS_SERVIDOR]
IP_ESCUCHA_SERVIDOR=127.0.0.1
PUERTO_ESCUCHA_SERVIDOR=23456
LONGITUD_BYTES_LECTURA=1024
NUMERO_MAXIMO_CLIENTES_ATENDER=15
ORDEN_FIN_TRANSMISION=#quit
 # Definir si el modo de funcionamiento del servidor será en modo
 # secuencial con una sola instancia, valor 1, o por el contrario
 # funcionará en modo paralelo creando un hijo, proceso fork, por cada
 # solicitud a atender

MODO_FUNCIONAMIENTO_SECUENCIAL=0
 # path donde guardar los mapas recibidos del servidor de mapas remoto
PATH_REPOSITORIO_DATOS_MAPAS=/mnt/filer-local/html/desarrollo/datos/MGM/mapas/yahoo_maps

[FICHERO_LOG]
 # Posibles valores (0,1)
ACTIVAR_FICHERO_LOG=1
PATH_FICHERO_LOG=/mnt/filer-local/html/desarrollo/datos/MGM/logs/servidor.log

[WEB_SERVER]
HOST_SERVIDOR_WEB_MAPAS=192.168.3.102
 # Directorio desde el cual el servidor web servirá los distintos mapas
DIRECTORIO_ROOT_SERVIDOR_WEB_MAPAS=maps

A continuación se detalla cada una de las secciones del fichero de configuración:

- *[DATOS_SERVIDOR]*
  En esta sección se definen los datos que gestionan las conexiones del servidor de comunicaciones, tales como la dirección IP y el puerto en la que escucha las peticiones de los dispositivos móviles, el número máximo de solicitudes a poner en espera en el caso de funcionamiento secuencial mientras se atiende a una petición, la longitud en bytes de lectura en el socket, y la orden de fin de trasmisión por parte del dispositivo móvil.
- *[FICHERO_LOG]*
  En esta sección se especificará si se quiere realizar un log que registre las acciones del servidor, y en caso afirmativo, el path absoluto o relativo del fichero log.
- *[WEB_SERVER]*
  En esta sección se detallan las opciones relacionadas con la respuesta y ubicación de los mapas de las solicitudes recibidas. Por un lado se especifica el host que identifica el servidor web que contiene los mapas descargados y el directorio en el que se encuentran, de tal forma que la respuesta que recibe un dispositivo móvil ante una solicitud, especifica la URL de acceso al mapa de la forma http://host/directorio/mapa.

La siguiente figura muestra el diagrama del flujo de datos y las acciones que se llevan a cabo en el servidor de comunicaciones.

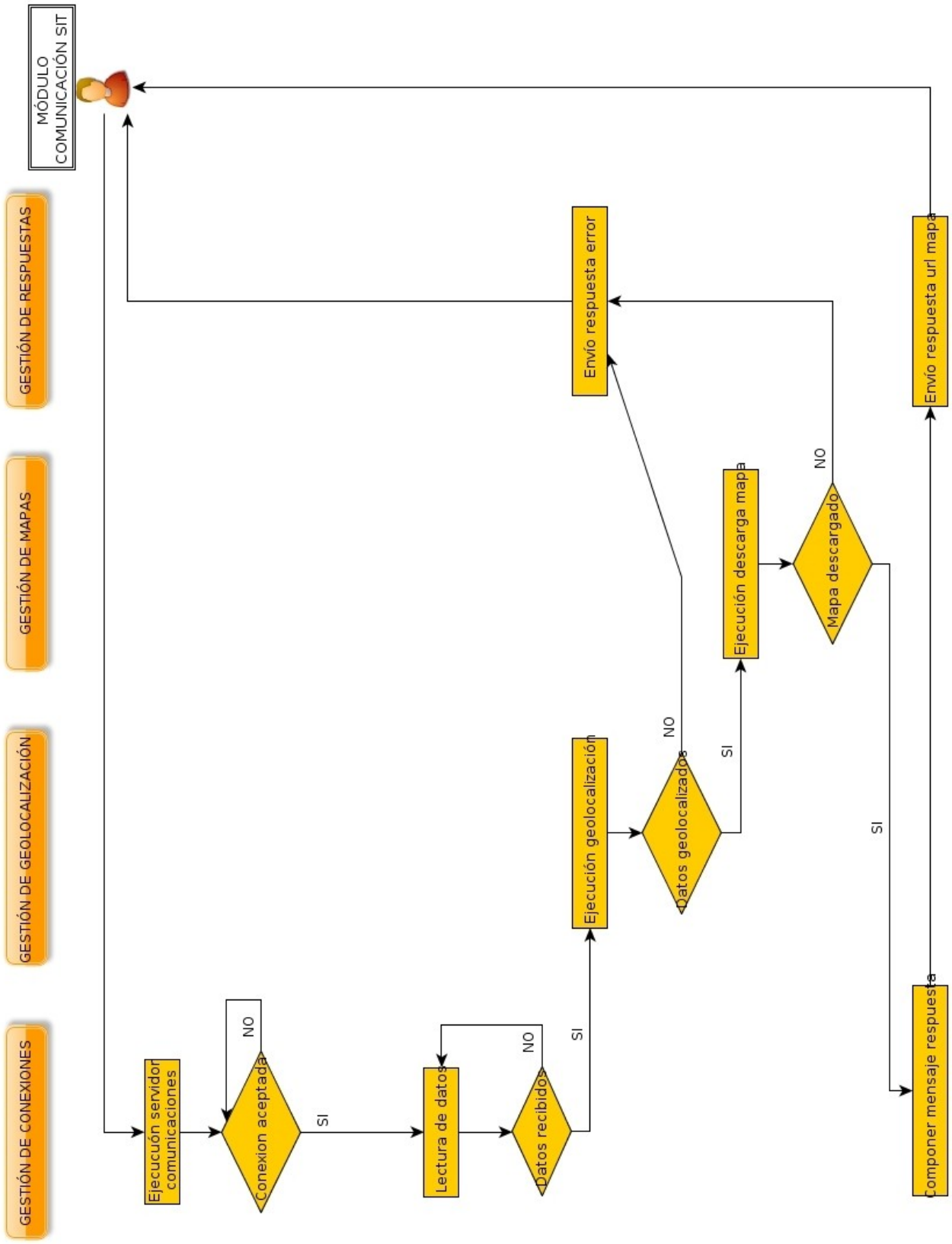

Figura 13: Diagrama del servidor de comunicaciones

5.3.1.c Servidor de geolocalizaciones

La tarea fundamental de este servidor, se centra en obtener y devolver la geolocalización de los datos que le proporciona el servidor de comunicaciones, procedentes de las peticiones de los dispositivos móviles del SIT. Hay que aclarar que el proceso de geolocalización es realmente una pseudo-geolocalización, ya que, para la ubicación especificada en la petición se obtiene su latitud y su longitud, en cambio no se calcula su altura, requisito necesario para la que geolocalización sea total.

El formato de la petición XML a geolocalizar debe ser un flujo de datos con codificacion UTF-8 con la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
    <localizacion>
        <direccion>Avenida de la universidad</direccion>
        <numero>5</numero>
        <ciudad>Leganes</ciudad>
        <provincia>Madrid</provincia>
        <localizador>[opcional]</localizador>
        <latitud>[opcional]</latitud>
        <longitud>[opcional]</longitud>
        <nivel_zoom>[opcional]</nivel_zoom>
    </localizacion>
```

A continuación se detalla el significado de cada una de las líneas:

- <?xml version="1.0" encoding="UTF-8"?>
  Etiqueta que indica el inicio del fichero xml, declarando su versión y su tipo de codificación. Son datos necesarios para la inicialización del parser del xml que del servidor de geolocalizaciones. Es necesario que el flujo de datos se encuentre en codificación UTF-8.
- *<localizacion>*
  Etiqueta que indica el inicio de los datos de la localización para la que se quiere obtener su geolocalización.
- *<direccion>Avenida de la universidad</direccion>*
  Etiqueta que nos indica la dirección (calle, avenida, plaza, etc) a geolocalizar. Solo incluye el nombre de la dirección y no el número ni la localidad ni provincia. Es necesario que el identificador de la dirección (calle, avenida, plaza) se especifique en su totalidad y no abreviada (c/, av., plza.).

- *<numero>5</numero>*

  Etiqueta que indica el número de la dirección anteriormente especificada.

- *<ciudad>Leganes</ciudad>*

  Etiqueta que indica la localidad de la dirección anteriormente especificada.

- *<provincia>Madrid</provincia>*

  Etiqueta que indica el nombre de la provincia de la dirección anteriormente especificada.

- *<codigo_postal>28911</codigo_postal>*

  Etiqueta que indica el código postal de la dirección anteriormente especificada.

- *<localizador></localizador>*

  Etiqueta que puede especificar la unión de la dirección, el número, la ciudad y la provincia. Si se especifica tendrá preferencia a la hora de realizar la geolocalización frente a las etiquetas anteriores.

- *<latitud></latitud>*

  Etiqueta que indica la latitud de la ubicación especificada o no. Puede darse el caso en el cual el usuario (cliente) sepa la geolocalización de la ubicación, para la cual quiere descargar el mapa. En este caso la petición realizada contendrá este valor y el geolocalizador no realizará la geolocalización, sino que simplemente parseará el *xml*, y si es correcto devolverá en la respuesta este dato junto con la longitud.

- *<longitud></longitud>*

  Etiqueta que indica la longitud de la ubicación especificada o no. Puede darse el caso en el cual el usuario (cliente) sepa la geolocalización de la ubicación, para la cual quiere descargar el mapa. En este caso la petición realizada contendrá este valor y el geolocalizador no realizará la geolocalización, sino que simplemente parseará el *xml*, y si es correcto devolverá en la respuesta este dato junto con la latitud.

- *<zoom></zoom>*

  Indica el nivel se zoom para el que se descargará la imagen. Hay 12 niveles de zoom

- *</localizacion>*

  Etiqueta que indica el fin de los datos a geolocalizar.

Al igual que el servidor de comunicaciones, existe una particularización o configuración de ciertas características del servidor a la hora de ejecutarse recogidas en un fichero de configuración (fichero

de texto con un formato determinado), incluido en el directorio de ficheros de configuración del MGM. El formato del fichero es el siguiente:

[DATOS_GEOLOCALIZADOR]

 # Yahoo Maps API key

 # Es necesario registrarse en Yahoo para obtener la clave.

*APPID=**********código API de Yahoo Maps**********

 # Host remoto del cual se obtendrán las geolocalizaciones de las peticiones

HOST_GEOLOCALIZADOR=http://local.yahooapis.com/MapsService/V1/geocode

 # Path absoluto fichero donde cachear las geoloalizaciones

PATH_ABSOLUTO_FICHERO_BERKELEY_DATOS_CACHEADOS_GEOLOCALIZACIONES=/mnt/filer-local/html/desarrollo/datos/MGM/geolocalizaciones/cache_geolocalizaciones.db

 # delimitador máximo del tiempo que permanecerán las geolocalizciones en la cache

 # Muy importante se mide en segundos

 # Actualmente el valor es 20 dias

 # Si se establece el valor a 0 la geolocalización nunca expirará

DELIMITADOR_TIEMPO_MAXIMO_CACHEO_GEOLOCALIZACIONES=1728000

[GRUPOS_DE_CAMPOS_EN_XML]

 # El valor de esta clave debe ser el nombre de grupo en donde especificaremos las

 # claves y valores de los tags del fichero xml

GRUPO_LOCALIZACION=localizacion

[TRADUCCION_CAMPOS_GRUPO_localizacion]

 # Cada clave es un atributo del objeto geolocalizador en el código fuente

 # Cada valor de la clave se corresponderá con una etiqueta del fichero xml de

 # las peticiones de geolocalización

 # Un ejemplo de fichero xml es el siguiente

 # <localizacion>

 # <localizador></localizador>

 # <direccion>Barrionuevo</direccion>

 # <numero>3</numero>

 # <ciudad>Leganes</ciudad>

 # <provincia>Madrid</provincia>

 # <codigo_postal>28911</codigo_postal>

 # <latitud>40.329018</latitud>

 # <longitud>-3.768280</longitud>

 # <nivel_zoom>6</nivel_zoom>
 # </localizacion>

DIRECCION=direccion
NUMERO=numero
CIUDAD=ciudad
PROVINCIA=provincia
LONGITUD=longitud
LATITUD=latitud
LOCALIZACION=localizador
CODIGO_POSTAL=codigo_postal

A continuación se el significado de cada una de las secciones del fichero de configuración:

- *[DATOS_GEOLOCALIZADOR]*
  En esta sección se establecen los datos principales del servidor de geolocalizaciones, tales como la clave de identificación para utilizar sel servicio web remoto de geolocalización de de Yahoo, el host remoto del servicio web de Yahoo para realizar las peticiones de geolocalización de ubicaciones, el path absoluto del fichero Berkeley DB que contendrá la base de datos de la cache de las geolocalizaciones, y el tiempo máximo en segundos de vigencia en cache de cada una de las geolocalizaciones.
- *[GRUPOS_DE_CAMPOS_EN_XML]*
  Sección que especifica cual es el nodo de primer nivel en la petición xml, que contendrá la definición de los campos o etiquetas del flujo de datos XML que conforman la ubicación a geolocalizar. Es útil si en un futuro se pretende realizar pruebas de cambio de formato en la petición XML especificando nuevos nombres en las etiquetas que conforman la petición.
- *[TRADUCCION_CAMPOS_GRUPO_localizacion]*
  Sección en la que se detalla la relación de los campos necesarios en el código fuente del servidor de geolocalizaciones con las etiquetas XML del fichero que conforma la petición.

  Permite abstraer completamente el significado de cado uno de las atributos de la clase con su nomenclatura en el XML que conforma la petición.

A la hora de resolver la geolocalización de una petición, se comprueba primero si existe en la cache de geolocalizaciones. Si existe, se comprueba si ha caducado. En el caso de que haya caducado o no exista en la cache, se realizará una petición al host remoto del servicio web de Yahoo, que devolverá

la geolocalización, y se insertará en la cache.

En caso de haber podido resolver la petición de geolocalización, el servidor de geolocalizaciones, devuelve el resultado con los datos de la geolocalización, al servidor de comunicaciones, en caso contrario devolverá el motivo por el que no se pudo resolver la petición de geolocalización.

La siguiente figura muestra el flujo de información y las acciones que se llevan a cabo en el servidor de geolocalizaciones.

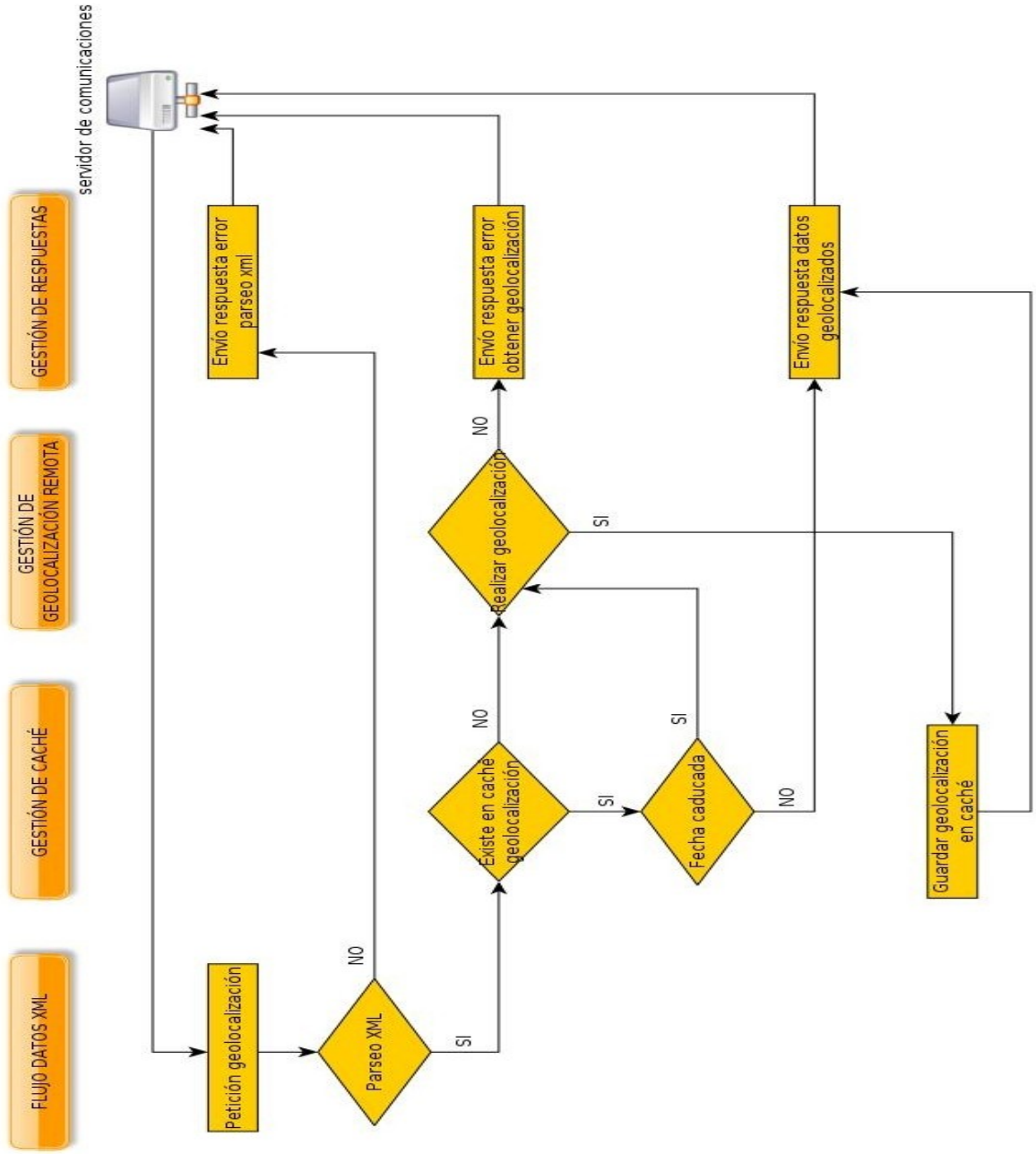

Figura 14: Diagrama del servidor de geolocalización

5.3.1.d Servidor de mapas

La tarea fundamental de este servidor consiste en descargar desde un host remoto del servicio web de Yahoo, la imagen del un mapa correspondiente a una geolocalización calculada por el servidor de geolocalizaciones y devolver como respuesta la ruta en la que se almacenó el archivo de la imagen.

La ruta en la que se se almacena el mapa se calcula aplicando una función de dispersión sobre los datos de geolocalización (longitud y latitud, zoom y tamaño de la imagen). De esta forma se evita almacenar todas las imágenes en un único directorio.

Si todas las imágenes estuviesen en un mismo directorio, el rendimiento del sistema de archivos sería muy bajo, al tener que manejar muchas entradas de mapas sobre un mismo directorio.

Al igual que los anteriores servidores, se pueden configurar ciertas funcionalidades del servidor mediante un archivo de configuración, que se carga nada más ejecutarse el servidor. A continuación se detalla el archivo de configuración para el servidor de mapas:

[DATOS_SERVIDOR_MAPAS]
 # Yahoo Maps API key. Necesaria para poder descargar mapas.
 # Necesario registrarse para obtener una key.
*APPID=**********código API de Yahoo Maps**********
 # Host remoto desde el que podremos descargar los mapas.
HOST_SERVIDOR_MAPAS=http://local.yahooapis.com/MapsService/V1/mapImage
 # Anchura por defecto en pixels de la imagen a descargar.
IMAGE_WIDTH=500
 # Altura por defecto en pixels de la imagen a descargar.
IMAGE_HEIGHT=620
 # Zoom por defecto de la imagen a descargar. Hay 12 niveles de zoom.
 # Por defecto si se omite este parámetro el valor es de 6.
IMAGE_ZOOM=1
 # Tipo por defecto de la imagen a descargar, puede ser png o gif.
 # Por defecto si se omite este parámetro la imagen se descarga en
 # formato png.
IMAGE_TYPE=gif
 # path donde guardar los mapas recibidos del servidor de mapas remoto.
PATH_REPOSITORIO_DATOS_MAPAS=/mnt/filer-local/html/desarrollo/docs/PFC/mapas/yahoo_maps
 # delimitador máximo del tiempo que permaneces las geolocalizciones el la cache berkeley

 # Muy importante se mide en segundos

 # Actualmente el valor es 20 días

 # Si se establece el valor a 0 la imagen no expirara

DELIMITADOR_TIEMPO_MAXIMO_CACHEO_IMAGENES=1728000

#DELIMITADOR_TIEMPO_MAXIMO_CACHEO_IMAGENES=0

La siguiente figura muestra el flujo de información y las acciones a llevar a cabo en el servidor de mapas

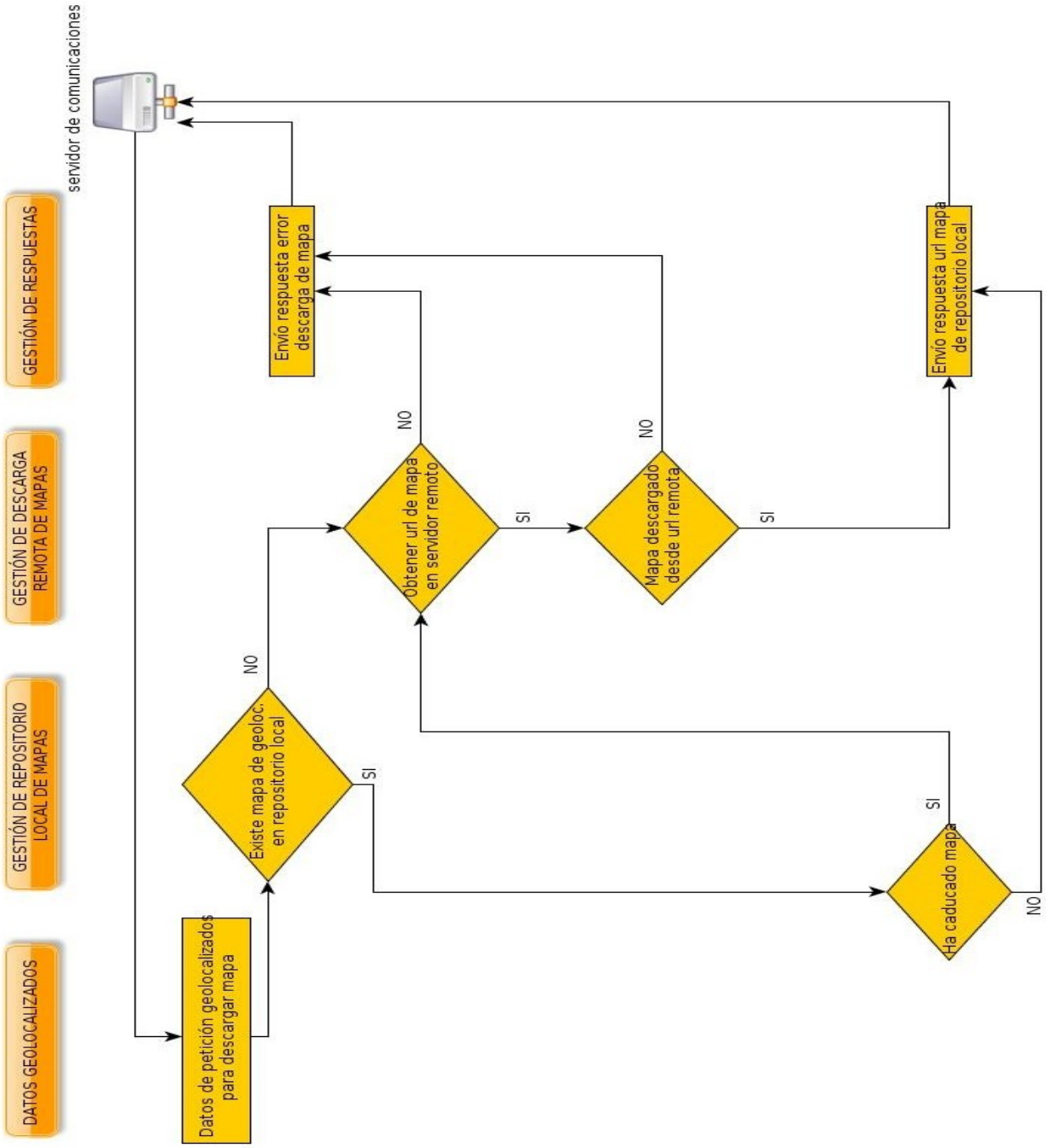

Figura 15: Diagrama servidor de mapas

5.3.1.e Servidor web Cherokee

Como su nombre indica, es un servidor web se encarga de resolver todas las peticiones por protocolo HTTP y devolver los mapas a los dispositivos móviles que los soliciten. El servidor de comunicaciones devuelve la respuesta o ubicación del mapa mediante una URL[29] con la estructura *protocolo://máquina/directorio/archivo_mapa* en la que el protocolo tiene el valor http, la máquina el nombre del host o máquina huésped en la que se instaló el servidor web, y el resto tanto el directorio como archivo en donde el servidor de mapas descargó la imagen del mapa.

Esta respuesta con la ubicación del mapa, es enviada al módulo de comunicaciones del SIT, que a su vez lo devolvera al dispositivo móvil al que le pertenezca la petición, y finalmente el dispositivo móvil solicitara al servidor web Cherokee el mapa.

La única restricción que hay que tener en cuenta a la hora de configurar el servidor web, es que su directorio base de contenidos, también denominado "document root", debe ser el directorio del sistema de ficheros al que se aplica la dispersión en el cual se ubican los subdirectorios donde se almacenan los mapas descargados por el servidor de mapas.

5.4 Detalle del proyecto

5.4.1 Diseño general del sistema

En este apartado se van a abordar con detalle cada una de los módulos o servidores que conforman el MGM, así como los distintos algoritmos, técnicas y rutinas empleadas en la implementación del mismo. Se razonarán las distintas decisiones adoptadas en determinados momentos a lo largo del diseño e implementación del sistema.

Dado que el paradigma de programación empleado en el desarrollo del proyecto se basa en la orientación a objetos, el diseño de los módulos se ha realizado mediante el uso de diagramas de clases. Éste nos mostrará la estructura de clases existente, las relaciones entre ellas, e incluirá también los atributos y métodos implementadas en cada una de las clases.

La implementación del MGM está formado por 8 clases relacionadas entre sí, existiendo relaciones de herencia entre algunas clases.

La relación de herencia presente en las clases, se debe a la abstracción que se puede realizar de los distintos mensajes que puede visualizar el sistema. Por mensaje se entiende cualquier texto que

pueda mandar a la salida estándar (por defecto la pantalla) cualquiera de los servidores presentes en el MGM.

Las clases del MGM son las siguientes:

- *c_servidor*: Clase principal del MGM, es la que crea un servidor de comunicaciones y se encarga de gestionar el flujo principal del sistema, recibiendo las peticiones del módulo de comunicaciones del SIT, instanciando las distintas clases restantes y respondiendo a las peticiones solicitadas.
- *c_geolocalizador*: Clase que se encarga de geolocalizar las peticiones recibidas por la clase *c_servidor.* Incorpora una *caché* de geolocalizaciones para aumentar el rendimiento del sistema.
- *c_servidor_mapas*: Clase cuyo cometido principal es la descarga de las imágenes de mapas correspondientes a las geolocalizaciones resueltas por la clase c_geolocalizador. Incorpora una caché para no tener que descargar constantemente los mismos mapas.
- *c_berkeley:* Clase que implementa una pequeña base de datos del tipo Berkeley DB, en la que se guardan las distintas geolocalizaciones de las peticiones atendidas por el MGM.
- *c_configuracion_base*: Clase que inicializa los distintos servidores dependiendo de ficheros de configuración.
- *c_fichero_log*: Clase que se encarga de guardar trazas en ficheros de log específicos, para posteriores comprobaciones por parte del administrador del sistema o de los desarrolladores.
- *c_validacion*: Clase que se encarga de implementar funciones de validación de tipos de datos, valores de variables, etc.
- *c_mensajes*: Clase que gestiona todos los mensajes que produce el MGM. Por mensaje se entiende cualquier cadena que se escupa por la salida estándar en sistemas l*inux*

La siguiente figura muestra las clases utilizadas en el implementación del MGM, y las acciones llevadas a cabo entre ellas, para el correcto funcionamiento del sistema.

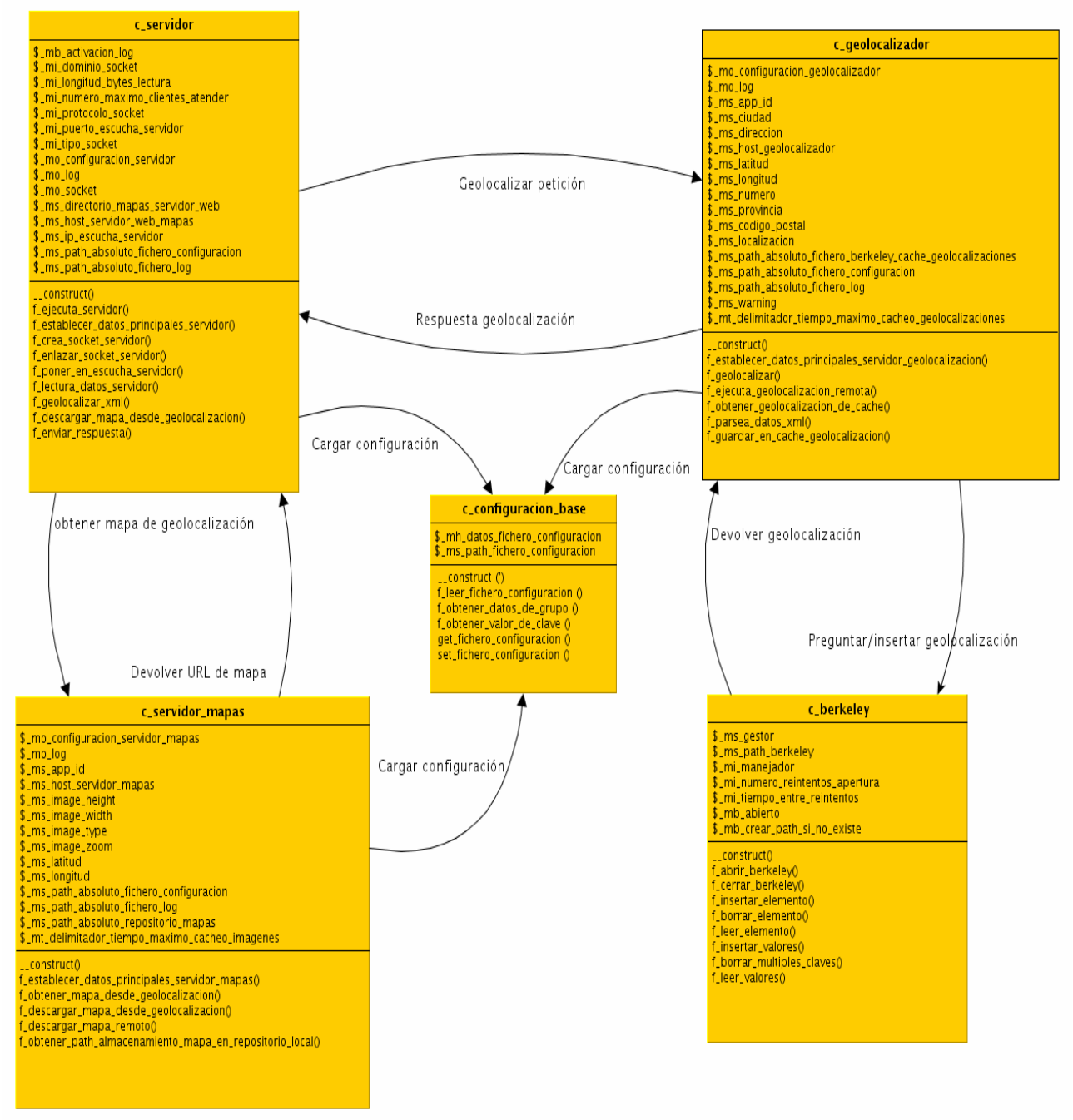

Figura 16: Diagrama de clases principales del MGM

5.4.2 Descripción de la clase c_servidor. Servidor de comunicaciones

Es la clase principal del MGM. Por un lado es la encargada de ejecutar un servidor de comunicaciones que atiende y responde a las distintas peticiones del los clientes, y por otro lado es la clase que gestiona el flujo principal del programa interactuando con los distintos servidores para

la correcta resolución de las peticiones recibidas.

Esta clase utiliza los siguientes módulos de PHP:

- Posix: para la creación de procesos hijos, si se configura el servidor en modo paralelo.
- Sockets: para la utilización de sockets como gestión de las comunicaciones.

A continuación se detallan los principales atributos y métodos de la clase. Si se quiere obtener un mayor detalle de la clase se puede localizar en la documentación del código fuente.

Los atributos principales de la clase son:

Atributo	Descripción
$_mb_activacion_log	Indica si la instancia de la clase escribirá trazas en el fichero log.
$_mi_dominio_socket	Dominio del socket, por defecto tiene el valor "*AF_INET*". Es el dominio necesario para comunicaciones por TCP/IP.
$_mi_longitud_bytes_lectura	Longitud de bytes por defecto en cada operación de lectura de datos por socket.
$_mi_numero_maximo_clientes_atender	Número máximo de clientes que se atenderán simultáneamente. En caso de sobrepasar su valor, los clientes de encolan y se atienden en orden se vayan liberando los clientes atendidos en curso.
$_mi_protocolo_socket	Protocolo de comunicación del socket. Por defecto tiene el valor "*SOL_TCP*" utilizado para comunicaciones por TCP/IP.
$_mi_puerto_escucha_servidor	Puerto de escucha del servidor por el que se atenderán las peticiones.
$_mi_tipo_socket	Tipo de socket. Por defecto tiene el valor "*SOCK_STREAM*" o socket de flujo de datos necesario en el tipo de comunicación que queremos.
$_mo_configuracion_servidor	Contiene una instancia u objeto de la clase *c_configuracion_base* con las definiciones necesarias para la correcta inicialización de la clase.
$_mo_log	Contiene una instancia de la clase *c_fichero_log* para escribir trazas y mensaje en el fichero log especificado en el objeto del atributo *$_mo_configuracion_servidor*.
$_mo_socket	Contiene un objeto nativo de PHP del tipo socket, utilizado para realizar todas las operaciones de gestión de comunicaciones con los clientes que realizan peticiones.
$_ms_directorio_mapas_servidor_web	Contiene el path absoluto del directorio en el que se ubican los mapas en el servidor web APACHE.
$_ms_host_servidor_web_mapas	Nombre del host del servidor web APACHE que contiene los mapas. Utilizado para devolver la URL del mapa como respuesta de las peticiones de los clientes.
$_ms_ip_escucha_servidor	Dirección IP en la que escucha las comunicaciones el servidor.
$_ms_path_absoluto_fichero_configuracion	Path absoluto del fichero de configuración que contiene las especificaciones con las que se inicializará la instancia del servidor de comunicaciones.
$_ms_path_absoluto_fichero_log	Path absoluto del fichero log en el que se grabarán los registros de las trazas del servidor de comunicaciones.

Los métodos principales de la clase se pueden resumir en:

Método	Descripción
__construct	Constructor de la clase.
f_ejecuta_servidor	Método que ejecuta el servidor. Es la primera función que se ejecuta tras el constructor de la clase.
f_establecer_datos_principales_servidor	Método que inicializa la clase con las especificaciones contenidas en el fichero de configuración.
f_crea_socket_servidor	Crear un extremo de una comunicación (socket) en el dominio especificado y del tipo indicado por variables miembro de la clase.
f_enlazar_socket_servidor	Enlazar un camino o una dirección de Internet a un conector (socket).
f_poner_en_escucha_servidor	El servidor se pone en escucha y para su ejecución hasta que recibe una conexión.
f_lectura_datos_servidor	Lectura de datos que envía un socket cliente Los datos se leen de forma binaria para que no haya problemas en diferentes plataformas o sistemas operativos con los retornos de carro y fin de línea.
f_geolocalizar_xml	Función que se encarga de geolocalizar un xml pasado como parámetro. Para ello realiza una petición al servidor de geolocalizaciones.
f_descargar_mapa_desde_geolocalizacion	A partir de unos datos de geolocalización, se obtiene la imagen correspondiente al mapa de la localización. Para ello realiza una petición al servidor de mapas.
f_enviar_respuesta	Envía la respuesta al socket ce la conexión del cliente (dicho socket sigue abierto)

A continuación se detalla en pseudocódigo las principales rutinas y procesos realizado por la clase que implementa el servidor:

5.4.2.a Función __construct

Constructor de la clase. Recibe como parámetro el path absoluto de un posible fichero de configuración para inicializar los atributos del servidor, con las directrices especificadas en este fichero. En el caso de no recibir un fichero de configuración, utilizará uno por defecto, que se encuentra ubicado en el directorio de configuración del servidor de comunicaciones.

Adicionalmente establece el dominio del socket como dominio de internet, el tipo de socket como *"stream"* y el protocolo como TCP, para la futura creación del socket de comunicaciones.

```
Inicio __construct
        instancia.path_absoluto_fichero_configuracion = param_fichero_configuracion
        instancia.dominio_socket = AF_INET
        instancia.tipo_socket = STREAM
        instancia.protocolo_socket =  TCP
Fin __construct
```

5.4.2.b Función f_ejecuta_servidor

Función principal del servidor de comunicaciones. Carga el fichero de configuración y establece los atributos de la clase a partir de los datos del fichero. Procede a la creación y puesta en escucha del

socket de comunicaciones y se ejecuta un bucle infinito, que escucha, trata y contesta peticiones. Es importante especificar que el servidor de comunicaciones una vez creado el socket se queda dormido hasta que haya alguna conexión entrante.

Una vez que el dispositivo móvil establece una conexión como resultado de una petición de mapa, el servidor de comunicaciones crea una instancia del servidor de geolocalizaciones, realizando una llamada a la función que geolocaliza los datos recibidos en la conexión. Una vez geolocalizada la petición, crea una instancia del servidor de mapas solicitando la descarga del mapa correspondiente a la geolocalización obtenida. Cuando el mapa ha sido descargado, envía la respuesta al servidor de comunicaciones del SIT. Si ocurriese cualquier error en los procesos anteriores la respuesta enviada indicaría el error y la imposibilidad de obtener el mapa de la petición solicitada.

Se debe tener en cuenta que si el servidor se configura para funcionar en modo paralelo, por cada petición de se crea un procese hijo que será el que atenderá esa petición.

```
Inicio  f_ejecuta_servidor
        obj_configuracion = f_carga_fichero_configuracion(instancia.path_absoluto_fichero_configuracion)
        f_establecer_datos_principales_servidor(obj_configuracion)
        // Creación de un socket por el que comunicarse con los dispositivos móviles
        instancia.socket = f_crea_socket_servidor()
        // Asociarse a una IP y un puerto
        f_enlazar_socket_servidor(instancia.socket)
        // Poner en escucha el socket
        f_poner_en_escucha_servidor(instancia.socket)
        while(1) //bucle infinito
                // En este punto el servidor se queda dormido hasta que haya conexiones entrantes
                obj_socket_disp_movil = socket_accept(instancia.socket)
                lanzar_hijo_si_funcionamiento_en_modo_concurrente_paralelo()
                si soy_padre
                        no_hacer_nada
                sino
                        datos_peticion_xml = f_lectura_datos_servidor( obj_socket_disp_movil)
                        resultado_geolocalizacion = f_geolocalizar_xml(datos_peticion_xml)
                        si falso == resultado_geolocalizacion
                                repuesta = 'No se pudo geolocalizar la peticion'
                                f_enviar_respuesta(obj_socket_disp_movil, respuesta)
                        sino
                                url_mapa = f_descargar_mapa_desde_geolocalizacion(resultado_geolocalizacion)
                                Si falso == url_mapa
                                        respuesta = ' No se pudo descargar el mapa de la peticion'
                                        f_enviar_respuesta(obj_socket_disp_movil, respuesta)
                                sino
                                        respuesta ='El mapa esta en la URL url_mapa'
                                        f_enviar_respuesta(obj_socket_disp_movil, respuesta)
                                fin si
                        fin si
                        socket_close(obj_socket_disp_movil)
                fin si
```

```
        fin_while
Fin f_ejecuta_servidor
```

5.4.2.c Función f_establecer_datos_principales_servidor

Su cometido principal es inicializar correctamente los atributos de la clase con los datos especificados en el fichero de configuración cargado. De esta forma, la dirección IP y el puerto por el que se mantendrán las conexiones con los dispositivos móviles, quedan definidos. También se definen atributos como el número de máximo de bytes en cada lectura del socket, así como el número máximo de clientes a atender y el path absoluto del fichero log en el que se almacenarán los mensajes y las trazas de depuración.

```
Inicio f_establecer_datos_principales_servidor
        instancia.ip_escucha_servidor = instancia.obj_configuracion.ip_escucha_servidor
        instancia.puerto_escucha_servidor = instancia.obj_configuracion.puerto_escucha_servidor
        instancia.numero_longitud_bytes_lectura = instancia.obj_configuracion.longitud_bytes_lectura
        instancia.num_clientes_atender = instancia.obj_configuracion.numero_maximo_clientes_atender
        instancia.path_absoluto_fichero_log = instancia.obj_configuracion.path_absoluto_fichero_log
Fin f_establecer_datos_principales_servidor
```

5.4.2.d Función f_crea_socket_servidor

Crea un socket por el que se establecerán todas las conexiones con el servidor de comunicaciones. En la creación del socket se reciben los parámetros de domino, tipo y protocolo. Para la realización del MGM se ha establecido:

- como dominio *AF_INET*: los protocolos de Internet IPv4/IPv6 y los protocolos TCP y UDP son los utilizado para esta familia.
- como tipo *SOCK_STREAM*: Proporciona flujos de bytes basados en una conexión bidireccional secuenciada, confiable. Se puede admitir un mecanismo de transmisión de datos fuera-de-banda.
- como protocolo *SOL_TCP*: Es un protocolo orientado a conexión, fiable, de flujo de datos en ambos sentidos. Garantiza que todos los paquetes serán recibidos en el mismo orden que fueron enviados. Si algún paquete se pierde durante la conexión, automáticamente se retransmite hasta que el destino no confirma su llegada.

```
Inicio f_crea_socket_servidor
        instancia.socket = socket_create(instancia.domino_socket, instancia.tipo_socket, instancia.protocolo)
        si instancia_socket = falso
                mensaje_error_log('Error al crear el socket')
```

```
                retornar falso
        sino retornar verdadero
        fin si
Fin f_crea_socket_servidor
```

5.4.2.e Función f_enlazar_socket_servidor

Esta función se encarga de enlazar el socket creado en la función *f_crea_socket_servidor* a una dirección IP y un puerto por el que escuchará las conexiones.

```
Inicio f_enlazar_socket_servidor
        si socket_bind(instancia.socket, instancia.ip_escucha, instancia.puerto_escucha) == falso
                mensaje_error_log('Error al enlazar el socket')
                sino retornar falso
        sino
                sino retornar verdadero
        fin si
Fin f_enlazar_socket_servidor
```

5.4.2.f Función f_poner_en_escucha_servidor

Esta función se encarga de poner en modo de escucha el socket anteriormente enlazado. Se especifica el número máximo de clientes a poner en la cola para atenderlos.

```
Inicio f_poner_en_escucha_servidor
        si socket_listen(instancia.socket, instancia.instancia.numero_maximo_clientes_atender) == falso
                mensaje_error_log('Error al poner en escucha el socket')
        fin si
Fin f_poner_en_escucha_servidor
```

5.4.2.g Función f_lectura_datos_servidor

Función que lee y devuelve los datos obtenidos del socket cliente pasado como parámetro. El tipo de datos es tratado como binario y no como texto. Si no se reciben datos salta un timeout por defecto del socket.

```
Inicio f_lectura_datos_servidor()
        retornar socket_read(obj_socket_cliente, instancia.longitud_bytes_lectura, tipo_lectura_raw);
Fin f_lectura_datos_servidor
```

5.4.2.h Función f_geolocalizar_xml

Esta función recibe como parámetro los datos de la petición del dispositivo móvil, crea un objeto del tipo geolocalizador (servidor de geolocalización) y comprueba la sintaxis el flujo de datos XML para comprobar que su sintaxis es correcta. Tras comprobar la sintaxis de los datos intenta

geolocalizarlos devolviendo el resultado de la geolocalización o falso en caso de error.

```
Inicio f_geolocalizar_xml
        obj_geolocalizador = new c_geolocalizador;
        obj_geolocalizador.inicializa_geolocalizador().
        datos_xml = obj_geolocalizador.f_parsea_datos_xml(peticion_cliente)
        si datos_xml == falso
                mensaje_error_log('Error al parsear los datos')
                retornar falso
        sino
                respuesta = obj_geolocalizador.f_geolocalizar(datos_xml)
                si respuesta == falso
                        mensaje_error_log('Error al geolocalizar los datos')
                        retornar falso
                sino
                        retornar respuesta
                fin si
        fin si
Fin f_geolocalizar_xml
```

5.4.2.i Función f_descargar_mapa_desde_geolocalizacion

Esta función recibe como parámetro los datos de una geolocalización para la que se quiere obtener la imagen del mapa. Se crea un objeto del tipo servidor_mapas (servidor de mapas) y se inicializa con la longitud, latitud y zoom de la imagen a obtener. Adicionalmente se calcula el path absoluto en el que se debe almacenar la imagen del mapa. A continuación de procede a su obtención (que no descarga si estuviese en la cache) y se devolvería falso en caso de no poder descargarse o en caso contrario el path absoluto en de la imagen obtenida, bien por descarga o por estar en la cache.

```
Inicio f_descargar_mapa_desde_geolocalizacion
        obj_servidor_mapas = new c_servidor_mapas;
        obj_servidor_mapas .inicializa_servidor_mapas()
        obj_servidor_mapas .set_latitud(datos_geolocalizacion.latitud)
        obj_servidor_mapas .set_longitud(datos_geolocalizacion.longitud)
        obj_servidor_mapas .set_zoom(datos_geolocalizacion.zoom)
        path_mapa = obj_servidor_mapas.f_calcula_path_mapa_almacenar
        si obj_servidor_mapas.f_obtener_mapa_desde_geolocalizacion(path_mapa) == falso
                mensaje_error_log('Error al descargar el mapa de la geolocalización')
                retornar falso
        sino
                devolver path_mapa
        fin si
Fin f_descargar_mapa_desde_geolocalizacion
```

5.4.2.j Función f_enviar_respuesta

Esta función recibe como parámetro la respuesta a enviar al cliente (URL del mapa, mensaje de error, …) y la envía por el socket abierto con el módulo de comunicaciones del SIT.

Se establece un número de reintentos para el que reenviar los datos en caso de error.

```
Inicio f_enviar_respuesta
        numero_reintentos = 6
        i = 1
        enviado = falso
        while (i < numero_reintentos && not enviado)
                enviado = socket_write(instancia.socket_cliente, respuesta)
                si enviado == false
                        num_reintentos ++
                fin si
        fin while
Fin f_enviar_respuesta
```

5.4.3 Descripción de la clase c_geolocalizador. Servidor de geolocalizaciones

Esta clase se encarga de realizar las geolocalizaciones de las peticiones atendidas que le suministra el servidor de comunicaciones. La geolocalización se realiza mediante llamadas a un host remoto que implementa un servicio de geolocalización, mediante el uso de un API específico. En el presente PFC, tal y como se indicó anteriormente el servicio y host corresponden a Yahoo Maps.

Dado que la resolución de las geolocalizaciones provenientes de las peticiones de los dispositivos móviles del SIT, debe ser lo más rápida y eficiente posible, este servidor incorpora una cache de geolocalizaciones de peticiones anteriormente resueltas.

A continuación se detallan los principales atributos y métodos de la clase. Si se quiere obtener un mayor detalle de la clase se puede localizar en la documentación del código fuente.

Los atributos principales de la clase son:

Atributo	Descripción
$_mo_configuracion_geolocalizador	Contiene una instancia u objeto de la clase *c_configuracion_base* con las definiciones necesarias para la correcta inicialización de la clase.
$_mo_log	Contiene una instancia de la clase *c_fichero_log* para escribir trazas y mensaje en el fichero log especificado en el objeto del atributo *$_mo_configuracion_geolocalizador*.
$_ms_app_id	Identificador de usuario para poder utilizar el API y los servicios remotos de *Yahoo Maps*.
$_ms_ciudad	Campo ciudad dentro de la definición XML que conforma una petición de geolocalización por parte del dispositivo móvil del SIT.
$_ms_direccion	Campo que identifica una calle, ciudad, avenida, etc, dentro de la definición XML que conforma una petición de geolocalización por parte del dispositivo móvil del SIT.
$_ms_host_geolocalizador	Nombre del host remoto que implementa el servicio de geolocalización por parte de *Yahoo Maps*.
$_ms_latitud	Campo que indica la latitud dentro de la definición XML que conforma una petición de descarga de mapas por parte del dispositivo móvil del SIT.
$_ms_longitud	Campo que indica la longitud dentro de la definición XML que conforma una petición

	de descarga de mapas por parte del dispositivo móvil del SIT.
$_ms_numero	Campo que indica el número de la localización indicada en el campo *$_ms_dirección,* dentro de la definición XML que conforma una petición de descarga de mapas por parte del dispositivo móvil del SIT.
$_ms_provincia	Campo que indica el nombre dentro de la definición XML que conforma una petición de descarga de mapas por parte del dispositivo móvil del SIT.
$_ms_codigo_postal	Campo que indica el código postal de la localización
$_ms_localizacion	Campo que indica la calle, número, ciudad y provincia dentro de la definición XML que conforma una petición de descarga de mapas por parte del dispositivo móvil del SIT.
$_ms_path_absoluto_fichero_berkeley_cache_geolocalizaciones	Contiene el path absoluto de la base de datos *Berkeley DB* que contiene la cache de las geolocalizaciones de peticiones anteriores.
$_ms_path_absoluto_fichero_configuracion	Path absoluto del fichero de configuración que contiene las especificaciones con las que se inicializará la instancia del servidor de geolocalizaciones.
$_ms_path_absoluto_fichero_log	Path absoluto del fichero log en el que se grabarán los registros de las trazas del servidor de geolocalizaciones.
$_ms_warning	Atributo que contiene una cadena con la razón por la que el servidor remoto no ha podido geolocalizar la petición solicitada.
$_mt_delimitador_tiempo_maximo_cacheo_geolocalizaciones	Número de segundos de vigencia de los registros de la *cache* de geolocalizaciones.

Los métodos principales de la clase se pueden resumir en:

Método	Descripción
__construct	Constructor de la clase.
f_establecer_datos_principales_servidor_geolocalizacion	Método que inicializa la clase con las especificaciones contenidas en el fichero de configuración.
f_geolocalizar	Método que obtiene el resultado de la geolocalización de una petición, ya sea de la *cache* o del servidor remoto de geolocalización.
f_ejecuta_geolocalizacion_remota	Método que realiza una petición de geolocalización al servidor remoto de geolocalización.
f_obtener_geolocalizacion_de_cache	Método que obtiene la geolocalización de una petición desde la *cache* de geolocalizaciones.
f_parsea_datos_xml	Método que parsea una estructura xml (petición del dispositivo móvil) para comprobar que está bien formada. Se invoca antes de realizar cualquier tipo de geolocalización.
f_guardar_en_cache_geolocalizacion	Método que guarda una geolocalización en el cache de geolocalizaciones.

5.4.3.a Función __construct

Constructor de la clase. Recibe como parámetro el path absoluto de un posible fichero de configuración para inicializar los atributos del servidor de geolocalizaciones, dependiendo de las

directrices almacenadas en el fichero de configuración. En caso de no recibir un fichero de configuración, utilizará el path absoluto de uno por defecto, que se encuentra ubicado en el directorio de configuración del servidor de geolocalizaciones.

Adicionalmente inicializa a vacío el atributo *_ms_warning* que indica la razón por la que no se ha podido geolocalizar la petición.

```
Inicio __construct
        instancia.path_absoluto_fichero_configuracion = param_fichero_configuracion
        instancia.ms_warning = ''
Fin __construct
```

5.4.3.b Función f_establecer_datos_principales_geolocalizador

Su cometido principal es inicializar correctamente los atributos de la clase con los datos especificados en el fichero de configuración cargado. De esta forma, el host remoto geolocalizador, la clave del API para utilizar los servicios del geolocalizador, el path absoluto del fichero cache de las geolocalizaciones resueltas, quedan definidos. También se definen atributos como tiempo máximo vigente para cada una de las geolocalizaciones cacheadas, así como el path absoluto del fichero log.

```
Inicio f_establecer_datos_principales_geolocalizador
        instancia.host_geolocalizador = instancia.obj_configuracion.host_geolocalizador
        instancia.app_id = instancia.obj_configuracion.app_id
        instancia.path_absoluto_fichero_berkeley_geolocalizaciones =
        instancia.obj_configuracion.path_absoluto_fichero_berkeley_geolocalizaciones
        instancia.delimitador_tiempo_maximo_cacheo_geolocalizaciones = instancia.obj_configuracion.
        delimitador_tiempo_maximo_cacheo_geolocalizaciones
        instancia.path_absoluto_fichero_log = instancia.obj_configuracion.path_absoluto_fichero_log
Fin f_establecer_datos_principales_geolocalizador
```

5.4.3.c Función f_parsea_datos_xml

Su cometido principal es comprobar la sintaxis de la petición. Para ello recibe como parámetro la cadena en formato XML proveniente del servidor de comunicaciones. Crea localmente un objeto xml y ejecuta la función de parseo (comprobación sintáctica) de los datos en el objeto creado. Si el parseo fue correcto, devuelve un hash (array asociativo) con los datos de la petición, en el que cada clave del hash se corresponde con un nodo de la petición xml, y el valor de clave del hash con el contenido del nodo.

En el caso de no poder parsear la petición devuelve *falso*, indicando que hubo un error de parseo, o lo que es lo mismo el *xml* no está bien formado.

```
Inicio f_parsea_datos_xml
        hash_datos_xml = array()
        obj_xml = xml_simple_load_string(param_peticion_xml)
        si obj_xml == falso
                retorno_falso
        fin si

        hash_datos_xml[direccion] = obj_xml.direccion
        hash_datos_xml[calle] = obj_xml.calle
        i hash_datos_xml[numero] = obj_xml.numero
        hash_datos_xml[ciudad] = obj_xml.ciudad
        hash_datos_xml[provincia] = obj_xml.provincia
        hash_datos_xml[longitud] = obj_xml.longitud
        hash_datos_xml[latitud] = obj_xml.latitud
        hsh_datos_xml[codigo_postal] = obj_xml.codigo_postal
        retorno hash_datos_xml;
Fin f_parsea_datos_xml
```

5.4.3.d Función f_geolocalizar

Su objetivo es devolver la geolocalización de un hash (array asociativo) pasado por parámetro, cuyos datos se corresponden con una petición de un dispositivo móvil.

Inicialmente se inicializan los atributos de la instancia de la clase, con los valores del hash. A continuación se distinguen 2 casos a la hora de geolocalizar los datos:

- Si el en el flujo de datos xml correspondiente a la petición no se especificó latitud y la longitud, entonces se intenta buscar la geolocalización de los datos en la caché. Si existe y no ha expirado se devuelve, en caso contrario se realiza una geolocalización al host remoto de Yahoo Maps, mediante el uso del API, y se devuelve el resultado de la geolocalización en caso de éxito o falso en caso de error en la geolocalización.

  En caso de que la geolocalización remota haya tenido éxito, se guarda en la cache.

- Si el dispositivo móvil, ha especificado tanto latitud como longitud, se comprueban esos datos en la cache. Si no se encuentran en la cache, se realiza una geolocalización remota y se devuelve su resultado.

  Si se encuentran en la cache y son iguales no se hace nada, si son distintos, se realiza una geolocalización remota.

  En caso de que la geolocalización remota haya tenido éxito, se guarda en la cache.

```
Inicio f_geolocalizar
        instancia.direccion = param_hash_datos[direccion]
        instancia.calle = param_hash_datos[calle]
        instancia.numero = param_hash_datos[numero]
```

```
instancia.ciudad = param_hash_datos[ciudad]
instancia.provincia = param_hash_datos[provincia]
instancia.longitud = param_hash_datos[longitud]
instancia.latitud = param_hash_datos[latitud]
instancia.codigo_postal = param_hash_datos[codigo_postal]

si (instancia.latitud == '' or instancia.longitud == '')
        // No han pasado en el xml ni latitud ni longitud
        si (datos_de_cache = intancia.obtener_geolocalizacion_de_cache()) == falso
                si instancia.obtener_geolocalizacion.remota() == falso
                        mensaje_error_log('Error al  obtener la geolocalización')
                        retorno falso
                sino
                        instancia.guardar_geolocalizacion_en_cache()
                fin si
        sino
                // Obtenido de cache
                instancia.longitud = datos_de_cache[longitud]
                instancia.latitud = datos_de_cache[latitud]
        fin si
else
        // han pasado una longitud y latitud, comprobar si es la correcta
        si (datos_de_cache = intancia.obtener_geolocalizacion_de_cache())
                si (instancia.longitud != datos_de_cache[longitud]) or
                   (intancia.latitud != datos_de_cache[latitud]
                        instancia.guardar_geolocalizacion_en_cache()
                fin si
        si no
                // no se encontró la geolocalización en la cache. Ejecutar geolocalización remota
                si instancia.obtener_geolocalizacion.remota() == falso
                        mensaje_error_log('Error al obtener la geolocalización')
                        retorno falso
                sino
                        instancia.guardar_geolocalizacion_en_cache()
                fin si
        fin si
fin si
retornar array(instancia.latitud, instancia.longitud)
```

Fin f_geolocalizar

5.4.3.e *Función f_ejecuta_geolocalizacion_remota*

Esta función realiza una petición al host remoto de Yahoo Maps que se encarga de devolver la geolocalización de las peticiones recibidas. Para ello, a partir de los atributos de la clase, compone la cadena de parámetros que conforman la URL de la petición de geolocalización.

El formato de la respuesta de la petición es una cadena con código PHP o XML, con lo que, evaluándolo directamente estará disponible en el código fuente. Tras realizar la petición se evalúa el resultado de la misma para comprobar si ha habido algún error o se inicializan los atributos de longitud y latitud de la instancia, con el resultado obtenido.

Además de inicializarse los campos de la clase con la respuesta, se retorna el hash de la misma

```
Inicio f_ejecuta_geolocalizacion_remota
        url_geolocalizacion_remota =  intancia.host_geolocalizador + http_build_query(instancia.app_id,
        instancia.localizacion, instancia.calle, instancia.numero, instancia.ciudad, instancia.provincia,
        instancia.codigo_postal ,  formato_resultado_php_o_xml_a_elegir)
        hash_respuesta = f_obtener_datos_geolocalizacion_de_servidor_remoto(url_geolocalizacion_remota)
        si hash_respuesta == falso
                mensaje_error_log('Error al  descargar el mapa de la geolocalización')
                retorno_falso
        fin si
        instancia.longitud = hash_respueta[longitud]
        instancia.latitud = hash_respuetsa[latitud]
        retorna hash_respuesta
Fin f_ejecuta_geolocalizacion_remota
```

5.4.3.f Función f_obtener_geolocalizacion_de_cache

Esta función, se encarga de devolver un hash con la geolocalización de los datos de la instancia que lo solicita. Para ello, calcula la clave a buscar en el fichero Berkeley de geolocalizaciones. La clave se calcula a partir de la unión de la calle, número, ciudad y provincia, todo ello normalizado, eliminando espacios y acentos sustituyéndolos por guiones bajos y el carácter sin acentuar.

El valor de un registro de la claves está formado por la cadena [longitud|latitud|timestamp_registro], que identifica la longitud, latitud y timestamp (fecha en segundos desde 1970) en el que se insertó o actualizó en la cache. Ese timestamp sirve para saber si ha caducado o no ese valor de geolocalización en la cache.

De este modo en la cache se almacenan registros del tipo:

[calle+numero+ciudad+provincia]= [longitud|latitud|timestamp_registro]

En caso de no encontrarse la clave o haber caducado se devuelve falso. En caso contrario se devuelve el hash con la longitud y latitud del registro.

```
Inicio f_obtener_geolocalizacion_de_cache
        clave_busqueda = instancia.f_calcula_clave_para_cache_geolocalizaciones()
        hash_geolocalizacion = __f_obtener_datos_de_berkeley(
                instancia,path_absoluto_fichero_cache_geolocalizaciones, clave_busqueda)
        si hash_geolocalizacion == falso
                retornar falso
        sino
             si f_ha_caducado_cache(hash_geolocalizacion, instancia.timepo_maximo_cacheo_geolocalizaciones)
                retornar falso
            sino
                return  hash_geolocalizacion
           fin si
        fin si
```

Fin f_obtener_geolocalizacion_de_cache

5.4.3.g Función f_guardar_en_cache_geolocalizacion

Esta función, se encarga de devolver y guardar los datos de la geolocalización actual de la instancia en el fichero Berkeley de cache de geolocalizciones. Para ello calculan tanto la clave como el valor a guardar y inserta la tupla clave-valor en en el Berkeley. En caso de no existir la clave se crea y si ya existía, se actualiza con el nuevo valor.

La estructura del registro es :

[calle+numero+ciudad+provincia]= [longitud|latitud|timestamp_registro]

La función devuelve verdadero o falso en caso de que ha haya guardado o no el registro en la cache.

```
Inicio f_guardar_en_cache_geolocalizacion
        hash_datos_guardar = instancia.f_calcula_hash_para_cache_geolocalizaciones()
        hash_erroneos = __f_inserta_datos_de_berkeley(
                instancia,path_absoluto_fichero_cache_geolocalizaciones, hash_datos_guardar)
        si hash_erroneos == falso
                mensaje_error_log('Error al guardar los datos en cache')
                retornar falso
        sino
             retornar verdadero
                fin si
Fin f_guardar_en_cache_geolocalizacion
```

5.4.4 Descripción de la clase c_servidor_mapas. Servidor de mapas

Esta clase es la encargada de gestionar todas las operaciones relacionadas con las imágenes de los mapas. Una vez que una petición de un dispositivo móvil ha sigo geolocalizada, el servidor de comunicaciones solicita al servidor de mapas la imagen del mapa correspondiente de dicha geolocalización.

La descarga del mapa se realiza mediante llamadas a un servicio web ubicado en un host remoto mediante el uso de un API específico. En el presente PFC, tal y como se indicó anteriormente el servicio y host remoto corresponden a Yahoo Maps.

La descarga de los distintos mapas provenientes de las peticiones debe ser lo más rápida y eficiente posible, este servidor incorpora una cache de imágenes de mapas, correspondientes a peticiones anteriormente resueltas.

A continuación se detallan los principales atributos y métodos de la clase. Si se quiere obtener un

mayor detalle de la clase se puede localizar en la documentación del código fuente.

Atributo	Descripción
$_mo_configuracion_servidor_mapas	Contiene una instancia u objeto de la clase *c_configuracion_base* con las definiciones necesarias para la correcta inicialización de la clase.
$_mo_log	Contiene una instancia de la clase *c_fichero_log* para escribir trazas y mensaje en el fichero log especificado en el objeto del atributo *$_mo_configuracion_servidor_mapas*.
$_ms_app_id	Identificador de usuario para poder utilizar el API y los servicios remotos de *Yahoo Maps*.
$_ms_host_servidor_mapas	Nombre del host remoto que implementa el servicio de descarga de mapas por parte de *Yahoo Maps*.
$_ms_image_height	Campo que indica la altura en píxeles de la imagen del mapa a descargar.
$_ms_image_width	Campo que indica la anchura en píxeles de la imagen del mapa a descargar.
$_ms_image_type	Campo que indica el tipo de imagen a descargar (gif, o jpg). Por defecto se usa el valor de gif
$_ms_image_zoom	Campo que indica el nivel de zoom de la imagen a descargar.
$_ms_latitud	Campo que indica la latitud dentro de la imagen del mapa a descargar.
$_ms_longitud	Campo que indica la longitud dentro de la imagen del mapa a descargar.
$_ms_path_absoluto_fichero_configuracion	Path absoluto del fichero de configuración que contiene las especificaciones con las que se inicializará la instancia del servidor de mapas.
$_ms_path_absoluto_fichero_log	Path absoluto del fichero log en el que se grabarán los registros de las trazas del servidor de mapas.
$_ms_path_absoluto_repositorio_mapas	Path absoluto del directorio en el que se almacearán los mapas.
$_mt_delimitador_tiempo_maximo_cacheo_imagenes	Número de segundos de vigencia de los mapas descargados.

Los métodos principales de la clase se pueden resumir en:

Método	Descripción
__construct	Constructor de la clase.
f_establecer_datos_principales_servidor_mapas	Método que inicializa la clase con las especificaciones contenidas en el fichero de configuración.
f_obtener_mapa_desde_geolocalizacion	Método que obtiene un mapa a partir de una geolocalización. En el caso de que el mapa no esté en *cache* lo descarga desde el servidor remoto de mapas.
f_descargar_mapa_desde_geolocalizacion	Método que se conecta con el hort remoto y descarga el mapa a partir de unos datos de geolocalización
f_descargar_mapa_remoto	Método que descarga un mapa identificado por una *URL* desde un servidor remoto de mapas.
f_obtener_path_almacenamiento_mapa_en_repositorio_local	Método que devuelve el path absoluto donde se almacena o se descargará un mapa a partir de los datos de su geolocalización y el zoom del mapa.

5.4.4.a Función __construct

Constructor de la clase. Recibe como parámetro el path absoluto de un posible fichero de configuración para inicializar los atributos del servidor de mapas, dependiendo de las opciones especificadas en el fichero de configuración. En caso de no recibir un fichero de configuración, utilizará uno por defecto, que se encuentra ubicado en el directorio de configuración del servidor de mapas.

```
Inicio __construct
        instancia.path_absoluto_fichero_configuracion = param_fichero_configuracion
        instancia.ms_warning = ''
Fin __construct
```

5.4.4.b Función f_establecer_datos_principales_servidor_mapas

Función que inicializa las propiedades principales por defecto de la instancia del servidor de mapas. Las propiedades de la clase son el host remoto del servidor de mapas, la clave o identificador para utilizar el API de Yahoo, el tipo de imagen por defecto del mapa (gif), la altura, anchura y zoom por defecto de la imagen del mapa a descargar, el tiempo máximo que puede permanecer en la cache el mapa, el path absoluto del directorio en el que se almacenarán los mapas y el path absoluto del fichero log en el que se escribirán trazas.

```
Inicio f_establecer_datos_principales_servidor_mapas
        instancia.host_servidor_mapas = instancia.obj_configuracion.host_servidor_mapas
        instancia.app_id = instancia.obj_configuracion.app_id
        instancia.image_type = instancia.obj_configuracion.image_type
        instancia.image_height = instancia.obj_configuracion.image_height
        instancia.image_width = instancia.obj_configuracion.width
        instancia.image_zoom = instancia.obj_configuracion.image_zoom

        instancia.delimitador_tiempo_maximo_cacheo_mapas =
        instancia.obj_configuracion.delimitador_tiempo_maximo_cacheo_mapas

        instancia.path_absoluto_fichero_log = instancia.obj_configuracion.path_absoluto_fichero_log
        instancia.path_absoluto_repositorio_mapas = instancia.obj_configuracion.path_absoluto_repositorio_mapas
Fin f_establecer_datos_principales_servidor_mapas
```

5.4.4.c Función f_obtener_path_almacenamiento_mapa_en_repositorio_local

Función que a partir del zoom, tipo de imagen, la longitud y la latitud de la instancia, es capaz de aplicar una función de dispersión hash y calcular un ruta en donde almacenar una imagen. La función hash se obtiene sumando cada ordinal de cada carácter de la cadena del identificador del mapa (zoom_longitud_latitud) y transformando a hexadecimal para después seleccionar el primer byte obteniendo de esta forma un numero que se corresponderá con el directorio

```
Inicio f_obtener_path_almacenamiento_mapa_en_repositorio_local
        nombre_imagen = instancia.image_zoom +' _' +  instancia.latitud + '_' + instancia.longitud
        suma_ordinal = 0
        for i = 0 to longitud(nombre_imagen)
                letra = nombre_imagen[i]
                suma_ordinal += ordinal(letra)
                i++
        fin for

        suma_hexadecimal = hexdec(suma_ordinal)
        // nos quedamos con el primer byte
        residuo = suma_hexadecimal && 0xFF
        path = instancia.path_absoluto_respositorio_mapas + '/' + residuo + '/' + nombre_imagen + '.' +
                instancia.image_type
        retornar path
Fin  f_obtener_path_almacenamiento_mapa_en_repositorio_local
```

5.4.4.d Función f_obtener_mapa_desde_geolocalizacion

Función que obtiene el mapa de la geolocalización especificada en los atributos de la clase y la almacena en el path absoluto pasado por parámetro. Para ello se le pasa un path en el que se descargará un mapa desde el servidor de mapas remoto. Un vez que se haya descargado el mapa, este vale como cache para futuros mapas a no ser que pase un cierto tiempo desde la última vez en la que se descargo el mapa. Esa horquilla de tiempo esta definida en el fichero de configuración. Si la variable miembro del objeto _mt_delimitador_tiempo_maximo_cacheo_imagenes es igual a cero entonces el archivo cacheado con la imagen del mapa nunca caduca, por lo que no se vuelve a descargar del servidor remoto.

```
Inicio f_obtener_mapa_desde_geolocalizacion
        si existe_fichero (param_path_absoluto_descarga_mapa)
                si f_ha_caducado_la_cache( param_path_absoluto_descarga_mapa=
                        retornar instancia.f_descarga_mapa_desde_geolocalizacion(
                                        param_path_absoluto_descarga_mapa)
                sino
                        retornar verdadero
                fin si
        sino
                retornar instancia.f_descarga_mapa_desde_geolocalizacion(
                                        param_path_absoluto_descarga_mapa)
        fin si
Fin  f_obtener_mapa_desde_geolocalizacion
```

5.4.4.e Función f_descargar_mapa_desde_geolocalizacion

Función que se encarga de conectarse con el host remoto de servidor de mapas, obtener el mapa desde una regionalización (indicada en los atributos de la instancia), descargar y guardar el mapa en un path absoluto pasado por parámetro. Si no se pasa un path absoluto, la función devuelve un error.

Para descargar el mapa, primero se realiza una petición al servidor remoto, desde el que se obtiene una respuesta a partir de la los datos solicitados correspondiente a la URL de la descarga del mapa (en el servidor remoto). Una vez obtenida esta URL se debe proceder a su descarga en el path indicado por parámetro.

```
Inicio f_descargar_mapa_desde_geolocalizacion

        hash_peticion_img = array()
        hash_peticion_img.appid = instancia.appid
        hash_peticion_img.latitud = instancia.latitud
        hash_peticion_img.longitud = instancia.longitud
        hash_peticion_img.image_type = instancia.image_type
        hash_peticion_img.image_height = instancia.image_height
        hash_peticion_img.image_width = instancia.image_width
        hash_peticion_img.image_zoom = instancia.image_zoom

        hash_parametros_peticion = http_build_query(hash_peticion_img)
        // Obtener la URL de la que descargar el mapa
        URL_descarga_imagen = f_obtener_URL_mapa_remoto(hash_peticion_img)
        si falso ==  URL_descarga_imagen
                retornar falso
        fin si

        si falso == f_descargar_mapa_remoto(URL_descarga_imagen, param_path_absoluto_descarga_img)
                retornar falso
        sino
                retornar   param_path_absoluto_descarga_img
        fin si
Fin  f_descargar_mapa_desde_geolocalizacion
```

5.4.4.f Función f_descargar_mapa_remoto

Descarga un archivo remoto especificado por un path (en formato URL) y lo almacena en un path de destino. Tanto la URL como el path de destino son parámetros de entrada de la función. Comprueba si existe el directorio de la descarga, y si no existe lo crea. Devuelve verdadero o falso, como resultado de la descarga.

```
Inicio f_descargar_mapa_remoto
        directorio_descarga = dirname(param_path_destino)
        si existe_directorio(directorio_descarga) == falso
                si crea_directorio(directorio_descarga) == falso
                        retornar falso
                fin si
        fin si
        si copiar(param_URL_descarga, param_path_destino)
                retornar verdadero
        sino
                retornar falso
        fin si
Fin  f_descargar_mapa_remoto
```

5.4.5 Descripción de la clase c_berkeley. Cache de geolocalizaciones

Esta clase es la encargada de gestionar todas las operaciones relacionadas con la cache de las geolocalizaciones.

La cache se compone de una base de datos empotrada del tipo Berkeley DB. El formato de la base de datos es de tipo árbol balanceado, que almacena pares de claves-datos. Por defecto el orden del árbol es lexicográfico, donde primero se localizan las claves más cortas y luego las más largas.

La base de datos físicamente se corresponde con un fichero, almacenado en el path especificado en el fichero de configuración del servidor de geolocalizaciones. Es necesario tener instalado en el sistema operativo las librerías necesarias para poder trabajar con este tipo de base de datos. En concreto, para el desarrollo del presente PFC, se ha utilizado la versión Berkeley DB v.4 (existen diferentes implementaciones del motor de la base de datos).

A continuación se detallan los principales atributos y métodos de la clase. Si se quiere obtener un mayor detalle de la clase se puede localizar en la documentación del código fuente.

Atributo	Descripción
$_ms_gestor	Tipo de gestor de la base de datos. Por defecto “db4” correspondiente a la versión 4.
$_ms_path_berkeley	Path absoluto del fichero que conforma la base de datos.
$_mi_manejador	Recurso que nos permitirá realizar todas las operaciones sobre la base de datos.
$_mi_numero_reintentos_apertura	Número de reintentos a la hora de realizar la apertura de la base de datos.
$_mi_tiempo_entre_reintentos	Tiempo que pasa entre cada reintento de apertura. Se mide en microsegundos.
$_mb_abierto	Booleano que indica si la base de datos ya ha sido abierta
$_mb_crear_path_si_no_existe	Booleano que indica se se crea el directorio que contiene la base de datos si éste no existe.

Los métodos principales de la clase se pueden resumir en:

Método	Descripción
__construct	Constructor de la clase.
f_abrir_berkeley	Método que abre una base de datos Berkeley DB. En caso de fallo de apertura, hay una serie de reintentos especificados en el constructor de la clase.
f_cerrar_berkeley	Método que cierra la base de datos Berkeley DB, asociada a la instancia de la clase.
f_insertar_elemento	Inserta o reemplaza un elemento de la base de datos ya abierta.
f_borrar_elemento	Elimina un elemento de la base de datos ya abierta.
f_leer_elemento	Devuelve el elemento especificado y su valor de una base de datos ya abierta.

f_insertar_valores	Método que inserta en una base de datos ya abierta un hash con las claves y valores asociados a las claves.
f_borrar_multiples_claves	Método que elimina un array de claves de la base de datos ya abierta.
f_leer_valores	Método que devuelve todas las claves y valores de una base de datos que se corresponden con un hash de claves especificado.

5.4.5.a Función __construct

Constructor de la clase. Recibe como parámetro el path absoluto del fichero Berkeley DB sobre el que operar, el gestor o driver de la base de datos, el número de reintentos y tiempo entre reintentos en la apertura de la base de datos, y finalmente si se crea el directorio del path del fichero Berkeley DB si éste no existe.

```
Inicio __construct
        instancia.path_berkeley = param_path_berkeley
        instancia.gestor = param_gestor
        instancia.numero_reintentos_apertura = param_numero_reintentos_apertura
        instancia.tiempo_entre_reintentos_apertura = param_tiempo_entre_reintentos_apertura
        instancia.crear_path_si_no_existe = param_crear_path_si_no_existe
        instancia.abierto = falso
Fin __construct
```

5.4.5.b Función f_abrir_berkeley

Función que se encarga de abrir una base de datos especificada por el la propiedad *path_berkeley* de la instancia. Se pasa por parámetro el modo en el que se abrirá el fichero o base de datos (modo lectura, escritura, creación). Devuelve un booleano indicado si se realizó la apertura.

```
Inicio f_abrir_berkeley
        si instancia.crear_path_si_no_existe and no_existe_directorio(instancia.path_berkeley)
                crear_directorio(instancia.path_berkeley)
        fin si

        contador = 1
        while (contador <= instancia.numero_reintentos_apertura) && (intancia.abierto = falso)
                obj_descriptor_fichero = dba_open(instancia.path_berkeley, param_apertura, instancia.gestor)
                si obj_descriptor_fichero == falso
                        // Nos se ha podido abrir. Reintentamos la apertura
                        sleep(instancia.tiempo_entre_reintentos_apertura)
                        contador ++
                sino
                        // Se ha abierto correctamente el fichero
                        instancia.abierto = verdadero
                        intancia.manejador = obj_descriptor_fichero
                        return verdadero
                fin si
        fin while
Fin f_abrir_berkeley
```

5.4.5.c Función f_cerrar_berkeley

Función que se encarga de cerrar la base de datos abierta, utilizada por el manejador de la clase.

```
Inicio f_cerrar_berkeley
        si instancia.abierto
                si dba_close(intancia.manejador)
                        instancia.abierto = falso
                fin si
        fin si

Fin f_cerrar_berkeley
```

5.4.5.d Función f_leer_elemento

Función que se encarga devolver el valor del elemento cuya clave concuerda en la base de datos con la especificada por parámetro. En caso de no encontrarse la clave devuelve falso.

```
Inicio f_leer_elemento
        return dba_fecth(param_clave, instancia.manejador)
        fin si
Fin f_leer_elemento
```

5.4.5.e Función f_insertar_elemento

Función que se encarga insertar o reemplazar en la base de datos, el elemento especificado con su clave y valor especificados por parámetro. Devuelve falso en caso de no haber podido insertar la clave-valor.

```
Inicio f_leer_elemento
        return dba_repalace(param_clave, param_valor, instancia.manejador)
        fin si
Fin f_leer_elemento
```

5.4.5.f Función f_borrar_elemento

Función que se encarga eliminar de la base de datos, el elemento especificado por su clave pasado por parámetro. Devuelve falso en caso de no haber podido eliminar el elemento de la base de datos, o que éste no exista.

```
Inicio f_borrar_elemento
        si dba_exists(param_clave, instancia.manejador)
                return dba_delete(param_clave,, instancia.manejador)
        sino
                return falso
        fin si
Fin f_borrar_elemento
```

5.4.5.g Función f_insertar_valores

Función que se encarga insertar de la base de datos, un *hash* de valores en el que se especifica la clave y el valor a insertar, por cada elemento del *hash*. Devuelve un *hash* con las claves y valores de los elementos que no se han podido insertar, o falso si no se ha podido acceder a la base de datos.

```
Inicio f_insertar_valores
        si instancia.f_abrir_berkeley(param_modo_creacion_escritura) == falso
                returno falso
        sino
                hash_erroneos = array()
                foreach (param_elementos as clave => valor)
                        si f_insertar_elemento(clave, valor) == falso
                                hash_erroneos[clave]=valor
                        fin si
                fin foreach
                f_cerrar_berkeley()
                retorno hash_erroneos
        fin si
Fin f_insertar_valores
```

5.4.5.h Función f_borrar_multiples_claves

Dado un array de nombres de claves, elimina esas claves del fichero Berkeley DB o base de datos. Devuelve un array con las claves que no se han podido insertar, o falso si no se ha podido acceder a la base de datos.

```
Inicio f_borrar_multiples_claves
        si instancia.f_abrir_berkeley(modo_lectura_escritura) == falso
                returno falso
        sino
                array_erroneos = array()
                foreach (param_claves as clave)
                        si f_borrar_elemento(clave, valor) == falso
                                array_erroneos.push(clave)
                        fin si
                fin foreach
                f_cerrar_berkeley()
                retorno array_erroneos
        fin si
Fin f_borrar_multiples_claves
```

5.4.5.i Función f_leer_valores

Dado un *hash* de entrada con valores identificados con "clave_a_buscar_en_berkeley" => "nombre_clave_devolver_en_resultado" busca en el Berkeley DB o base de datos esas claves y las devuelve en un *hash* de resultado.

Si cualquiera de las claves a buscar no existe o no ha podido ser recuperada del Berkeley DB se

devuelve la clave en el *hash* de valores devueltos con valor falso (booleano). Si no se puedo acceder al Berkeley Db o base de datos se devuelve falso.

```
Inicio f_leer_valores
        si instancia.f_abrir_berkeley(modo_lectura) == falso
                returno falso
        sino
                hash_devueltos = array()
                foreach (param_elementos as clave => clave_devolver)
                        // Si la clave no existe se devuelve falso
                        hash_devueltos[clave_devolver] = f_leer_elemento(clave)
                fin foreach
                f_cerrar_berkeley()
                retorno hash_devueltos
        fin si
Fin f_leer_valores
```

6. RESULTADOS

6.1.1 Resultados generales

El resultado principal del presente proyecto final de carrera, ha consistido en el análisis, desarrollo e implementación de un módulo de gestión de mapas (MGM), que se engloba dentro de un sistema de gestión turística (SIT) descrito en el apartado 1.1 Descripción del sistema de información turística

El MGM se divide a su vez en tres partes claramente diferenciadas:

- servidor de comunicaciones, que se encarga de establecer las conexiones con el módulo de comunicaciones del SIT (que previamente ha recibido las peticiones de mapas de los dispositivos móviles), para poder atender y contestar a las distintas peticiones.
- Servidor de geolocalización de las peticiones solicitadas al servidor de comunicaciones. Básicamente, a partir de unos datos de localización se conecta a un servidor remoto y obtiene sus datos geolocalizados (latitud y longitud), requisitos indispensables para poder descargar el mapa correspondiente.
- Servidor de mapas, que descarga desde un servidor remoto el mapa correspondiente a una geolocalización especificada.

Como ya se había comentado en el apartado 1.2 Objetivos, el principal objetivo a cumplir era la comunicación y descarga de mapas de ciertas localizaciones, correspondientes a una serie de solicitudes procedentes del módulo de comunicaciones del SIT, así como una serie de objetivos secundarios tales como:

- El MGM y el módulo de gestión de comunicaciones del SIT deben interactuar mediante el uso de sockets.
- Minimizar en todo lo posible el ancho de banda entre las comunicaciones para que el impacto económico, derivado del uso de protocolos de comunicación con coste asociado sea el mínimo.
- Usar algún tipo de memoria intermedia a modo de caché para evitar tener que volver a geolocalizar y servir mapas de solicitudes ya atendidas anteriormente disminuyendo los tiempos de respuesta de las solicitudes.

- Dotar al sistema de cierta escalabilidad ante posibles aumentos de demandas de solicitudes en un futuro.
- El MGM debe ser tolerante a fallos de solicitudes.
- Diseñar el MGM de tal forma que en un futuro se le puedan añadir nuevas funcionalidades necesarias para atender a nuevos objetivos.

A continuación se comprobará si se han cumplido los objetivos marcados al principio del proyecto:

Para poder cumplir el objetivo primordial, que permite la comunicación entre el MGM y el módulo de comunicaciones del SIT, así como la resolución de las peticiones de descargas de mapas solicitados por éste último, se ha implementado tanto el servidor de comunicaciones, como el servidor de geolocalización y de mapas, así como la puesta en escena de un servidor web que facilite el acceso a los mapas solicitados.

Se ha implementado un servidor de sockets en el servidor de comunicaciones del MGM, para poder comunicarse con el módulo de comunicaciones del SIT. Además se permite la configuración del mismo para que se pueda aplicar una política de atención de solicitudes de forma secuencial o en paralelo, aumentando el rendimiento del sistema en el último caso.

Para conseguir minimizar el ancho de banda se ha optado por un diseño de tipo de solicitud basado en un fichero con formato XML con los campos imprescindibles, así como la elección del protocolo TCP/IP y una implementación de sockets rigurosa en la parte del servidor de comunicaciones.

Adicionalmente, la utilización de un tipo de caché tanto en las geolocalizaciones como en los mapas descargados, permite minimizar el ancho de banda, ahorrando posibles peticiones ya resueltas, a la hora de geolocalizar y descargar mapas, y disminuyendo el tiempo de respuesta de las solicitudes cumpliendo así un objetivo secundario.

Para garantizar una correcta escalabilidad del MGM, su diseño modular permite poder crear tantas instancias de los distintos servidores del MGM (servidor de comunicaciones, servidor de geolocalización y servidor de mapas), además de los servidores web que resuelven las peticiones de las imágenes, en una o tantas máquinas como se quiera, permitiendo incluso la posibilidad de tener recursos compartidos, tales como las memorias caches de geolocalizaciones, los repositorios de mapas y los servidores web. Se puede tener por ejemplo, tantos servidores de comunicaciones como se quiera, en distintas máquinas que cachean los datos en una o varias caches, además de descargar

los mapas a un solo repositorio, accesible desde distintos servidores web en varias máquinas.

En el capítulo 7 ESCALABILIDAD DEL MGM se detalla como se puede alcanzar esa escalabilidad.

6.2 Pruebas

Para verificar que el los objetivos han sido cumplidos, se ha sometido a una serie de pruebas a los tres servidores del MGM y al servidor web.

Todas las pruebas se han realizado en un ordenador con las siguientes especificaciones:

- Portátil Dell Precision M65
- Sistema operativo: Ubuntu linux versión 10.04 (Karmic Koala)
- Memoria Ram: 2Gb
- CPU: Intel(R) Core(TM)2 CPU T7400 @ 2.16GHz
- Disco Duro de 160 Gb a 6400 rpm.
- Router ADSL con conexión de 6Mbs de velocidad a internet
- Enlace de Red por tecnología inalámbrica con tecnología WIFI funcionando a 54 Mbs.

6.2.1 Pruebas de petición y respuesta de solicitudes de mapas de ubicaciones

Para saber el tiempo de resolución y respuesta a las solicitudes de los dispositivos móviles se ha diseñado un plan de pruebas dividido en dos partes:

Por una lado se ha diseñado una interfaz gráfica, ejecutable desde cualquier navegador web. Se corresponde con un archivo HTML que contiene código PHP, mostrando un formulario en el que se se puede simular una petición de un dispositivo móvil mediante un fichero con el formato XML petición, o bien introduciendo la información necesaria en las cajas de texto específicas.

Una vez introducidos los datos el usuario podrá obtener la geolocalización de la ubicación indicada y posteriormente el mapa correspondiente a dicha geolocalización, mostrándose en el navegador.

Por otro lado, existe la opción de simular las peticiones de los dispositivos móviles del SIT, por medio de la consola del sistema. De este modo se da la posibilidad de ejecutar una serie de

comandos que ejecutan librerías especificas implementadas en PHP, simulando distintas peticiones de los dispositivos móviles. Para ello se ha implementado un cliente que simula el módulo de comunicaciones del SIT, abriendo un socket TCP/IP contra el servidor de comunicaciones del MGM, realizando una petición específica o varias correspondientes a un fichero XML de pruebas.

Cabe destacar que en la implementación de esta simulación el cliente puede actuar de forma secuencial lanzando peticiones y no ejecutando la siguiente petición hasta que no haya terminado la anterior, o por el contrario puede funcionar de forma paralela lanzando una serie de hijos o peticiones simultáneas que se resuelven independientemente del orden en que fueron lanzadas.

Independientemente de las opciones anteriores, se han realizado pruebas de rendimiento en el MGM dada sus múltiples opciones de configuración, por un lado activando o desactivando caches tanto en el servidor de geolocalización, como en el servidor de mapas, así como las opciones de configuración de forma secuencial o paralela en el servidor de comunicaciones, para atender las peticiones del módulo de comunicaciones del SIT.

A continuación se detalla en una tabla las pruebas realizadas por consola, simulando la geolocalización y descarga de mapas de 11 peticiones, activando y desactivando caches y funcionando en modo secuencial o paralelo a la hora de atender las peticiones.

Hay que tener en cuenta que el número de peticiones a tratar son 11, es decir, se simulan 11 solicitudes desde el módulo de comunicaciones del SIT, cada una de ellas correspondiente a un dispositivo móvil. Los dispositivos móviles pueden actuar de forma secuencial realizando cada uno de ellos una petición uno detrás de otro, o simultánea, realizando cinco de ellos una petición al mismo tiempo y luego cuando se vayan resolviendo los otros restantes realizan su petición, pero con la restricción de que sólo puede haber cinco a la vez.

La conexión de ADSL utilizada en el uso de servicios web remotos para obtener las geolocalizaciones y descargas de mapas, se corresponde con una conexión de 6Mb, teniendo en cuenta que la comunicación entre el router y el equipo en el que se realizan las pruebas se establece mediante WIFI.

En primer lugar se detalla la tabla para el servidor de comunicaciones funcionado en forma secuencial, es decir, atendiendo una petición detrás de otra en orden de llegada, y no se trata la siguiente petición hasta que no se haya finalizado el procesamiento de la petición en curso.

Modo secuencial en servidor de comunicaciones		
Datos	**Peticiones secuenciales**	**Peticiones simultáneas (5)**
caché de geolocalización NO caché de mapas NO	33,053 seg.	23,32 seg.
caché de geolocalización SI caché de mapas NO	19,22 seg.	11,76 seg.
caché de geolocalización SI caché de mapas SI	0,13 seg.	0,13 seg.

Tabla 3: Pruebas modo secuencial para 11 peticiones de datos

A continuación se detalla la tabla para el modo paralelo en el servidor de comunicaciones; esto significa que por cada petición recibida el servidor lanza un hijo que atiende dicha petición, dejando recursos disponibles para que puedan atenderse otras peticiones simultáneamente, mediante la creación de hijos adicionales.

Modo paralelo en servidor de comunicaciones		
Datos	**Peticiones secuenciales**	**Peticiones simultáneas (5)**
caché de geolocalización NO caché de mapas NO	32,15 seg.	22,50 seg.
caché de geolocalización SI caché de mapas NO	17,04 seg.	11,10 seg.
caché de geolocalización SI caché de mapas SI	0,11 seg.	0,11 seg.

Tabla 4: Pruebas modo paralelo para 11 peticiones de datos

De los resultados anteriores, se puede afirmar que la utilización de caches tanto en el servidor de geolocalizaciones como en el servidor de mapas, afecta de forma muy directa al rendimiento del sistema, reduciendo considerablemente el tiempo de respuesta de las peticiones de los distintos dispositivos móviles, siendo muy aconsejable su uso.

También se puede observar un menor tiempo de respuesta de las peticiones en modo paralelo independientemente del uso de caches, que se irá notando más a medida que el número de dispositivos móviles que realizan peticiones simultáneas aumente.

6.2.1.a Pruebas de rendimiento de caches. Berkeley DB vs Mysql

Tal y como se ha descrito anteriormente, el uso de memoria cache incrementa de forma muy significativa el rendimiento del sistema, disminuyendo considerablemente los tiempos de respuesta de las peticiones de los dispositivos móviles.

Existen muchas opciones para implementar memorias caches, pero claramente Berkeley DB se ha impuesto por su baja carga del sistema y sus imbatibles promedios y tiempos en operaciones de lectura y escritura frente a sistemas de bases de datos tradicionales avanzados, rápidos y ligeros como Mysql.

A continuación se detallan dos escenarios de prueba en cache, correspondientes a la inserción y actualización en masa de datos y la lectura de registros, calculando tiempos de respuesta medidos en segundos.

6.2.1.b Escenario 1. Rendimiento en la inserción y actualización de registros.

Para la realización de esas pruebas se ha tenido en cuenta el tiempo que se tarda en insertar secuencialmente varios registros tanto en el fichero Berkeley DB como en la base de datos Mysql. El tiempo se mide en segundos. Cuanto más bajo sea el valor más rápido es la cache.

Registros	Mysql	Berkeley
10	0,040	**0,064**
100	0,033	**0,068**
1.000	0,092	**0,149**
10.000	0,595	0,204
100.000	5,358	1,602
500.000	28,182	8,596
1.000.000	53,238	18,668

Figura 18: Inserción de registros

Registros	Mysql	Berkeley
10	0,083	0,0128
100	0,036	0,010
1.000	0,098	0,022
10.000	0,624	0,14
100.000	6,052	1,593
500.000	30,782	8,68
1.000.000	61,942	32,41

Figura 17: Actualización de registros

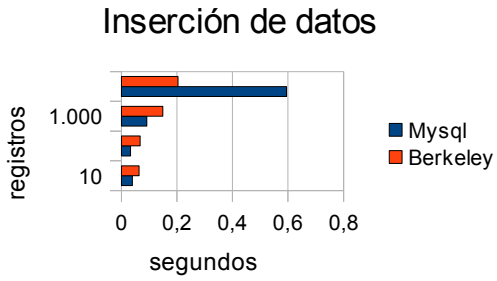

Figura 20: Gráfica inserción 2/2

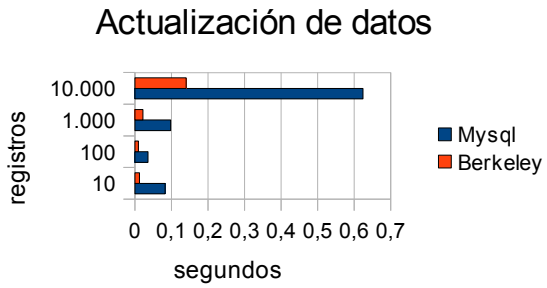

Figura 19: Gráfica de actualización 1/2

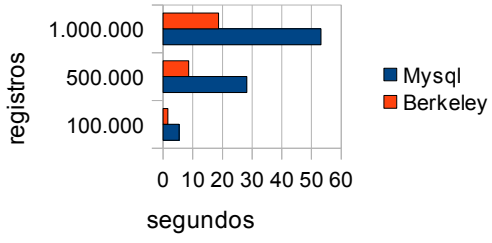

Figura 22: Gráfica inserción 2/2

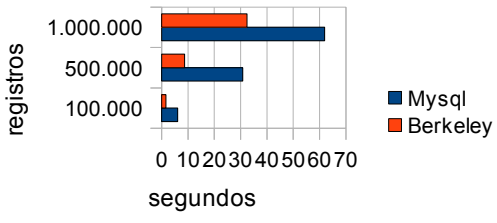

Figura 23: Gráfica de actualización 2/2

Claramente se puede observar cómo el rendimiento de Berkeley DB es muy superior al de Mysql, sobre todo a medida que se insertan o actualizan grandes cantidades de datos.

6.2.1.c Escenario 2. Rendimiento en la obtención de datos

Para la realización de esas pruebas se ha tenido en cuenta el tiempo que se tarda en leer y obtener secuencialmente varios registros tanto en el fichero Berkeley DB como en la base de datos Mysql. El tiempo se mide en segundos.

Registros	Mysql	Berkeley
10	0,020	0,00081
100	0,03	0,001
1.000	0,112	0,00728
10.000	0,0884	0,00955
100.000	8,864	0,802
500.000	45,007	6,478
1.000.000	91,512	13,57

Figura 25: Obtención de registros

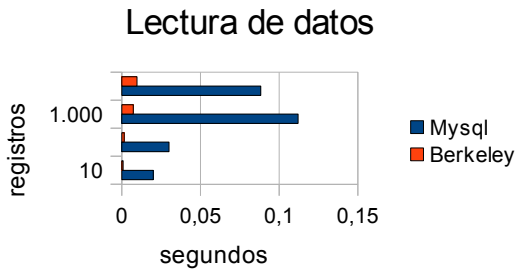

Figura 24: gráfica de obtención 1/2

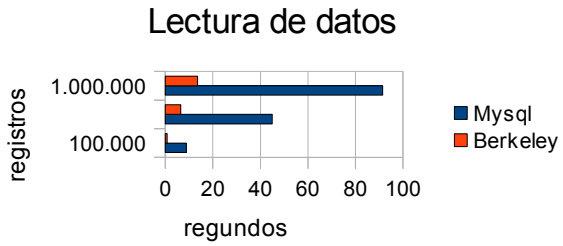

Figura 26: Gráfica de obtención 2/2

Claramente se puede observar cómo el rendimiento de Berkeley DB es exponencialmente superior frente a Mysql, indistintamente a la hora de obtener pequeñas o grandes cantidades de datos.

6.2.1.d Pruebas de rendimiento de servidores web

Se han realizado adicionalmente una serie de pruebas, para detectar el rendimiento entre un servidor en el servidor web Cherokee y un servidor web Apache, a la hora de poder servir las imágenes de los mapas, para poder cuantificar y contabilizar las peticiones por segundo que son capaces de atender.

Para ello se ha utilizado la herramienta de línea de comandos de linux “ab” (apache benchmark), que nos permite simular un número de peticiones elegido, sobre una URL, en este caso correspondiente a una dirección del servidor web que alojará un mapa del repositorio de mapas

descargados.

La prueba mide las peticiones por segundo que puede llegar a ofrecer el servidor, dependiendo directamente del número de peticiones que se le solicitan.

Para un número determinado de peticiones, se contabiliza los milisegundos que se tarda en procesarlas. Se calcula así el número de peticiones que se ejecutan en un segundo a partir del número de peticiones seleccionadas y los milisegundos empleados.

Cuanto más grande el el número de peticiones por segundo, más eficiente y rápido es el servidor web.

Num. Peticiones	Apache	Cherokee
400	2.859,84	3.528,89
800	3.062,63	3.509,48
1.000	3.133,71	3.617,76
10.000	3.690,52	5.145,16

Figura 28: peticiones por segundo

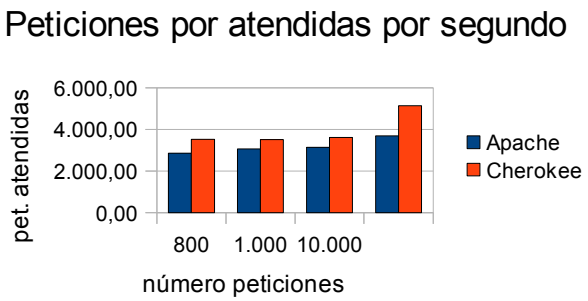

Figura 27: gráfica de peticiones por segundo

Claramente se puede observar que el servidor web Cherokee es más rápido que el servidor web Apache.

6.2.2 Otras consideraciones

Además de la implementación de un script que simule las peticiones de los dispositivos móviles por medio del protocolo TCP/IP bajo sockets, se ha tenido que generar cientos de ficheros para poder simular contenidos de peticiones.

Para ello se han obtenidos los puntos de interés (POIS) del software de navegación GPS Tom Tom Navigator, y se han transformado mediante herramientas y scripts implementados en ficheros xml con los formatos de las peticiones del SIT. De esta manera, se dispone de una base de datos y batería de pruebas para el MGM.

7. ESCALABILIDAD DEL MGM

Para conseguir una escalabilidad adecuada ante un posible aumento en la demanda de solicitudes o peticiones de descargas de mapas, provocadas por ejemplo por el aumento de dispositivos móviles, se debe tener en cuenta los posibles cuellos de botella que se puedan encontrar en el MGM. Para ello se evalúa la escalabilidad en varias partes del sistema:

- Servidor de comunicaciones: basado en sockets, puede, ante una demanda muy elevada de peticiones por parte de infinidad de dispositivos móviles, empezar a ralentizase, aunque tiene muchas opciones de configuración tales como el funcionamiento secuencial o paralelo mediante hijos. En este hipotético caso, la escalabilidad de esta parte es tan simple como crear otro servidor de comunicaciones en la misma u otra máquina en otro puerto, de este modo se puede tener tantos servidores como se desee. Tan sólo hay que copiar la estructura de librerías y fichero de configuración que conforman el MGM a otra máquina y modificar los ficheros de configuración, indicando el puerto de escucha.
- Servidor de geolocalizaciones: en el hipotético caso de que el acceso al sistema de cache (fichero Berkeley DB) sea muy alto con múltiples conexiones concurrentes, existe la posibilidad de poder configurar dónde se encuentra su fichero de cache de datos por cada servidor de comunicaciones del MGM, pudiendo especificarse distintos ficheros para distintos servidores, dividendo el acceso a los mismos disminuyendo de esta forma la carga del sistema. Para ello se especifica en el fichero de configuración del servidor de geolocalizaciones la nueva ruta correspondiente al fichero Berkeley DB que almacenará la cache del servidor en cuestión.
- Servidor de mapas: al igual que en el punto anterior, en el caso de que existan demasiadas peticiones, se puede establecer en una o varias máquinas, tantos servidores de mapas como se quiera.
- En el caso de servidor web no sea capaz de poder atender tantas peticiones, se pueden crear tantos servidores como se quiera, en los cuales se establece el mismo path absoluto de directorio en el que se almacena toda la estructura de las imágenes de los mapas. Tan solo se debe modificar en el archivo de configuración de un servidor de comunicaciones cualquiera, el nombre del servidor host que contendrá el servidor web, para atender las peticiones.

Otra solución más compleja pasa por tener un balanceador de carga a modo de frontal de peticiones web por parte de los distintos dispositivos móviles, y los servidores web que se estime oportuno, a los que periódicamente el balanceador frontal les realizará las peticiones. Está solución es muy rápida y eficaz y no es muy costosa a nivel de hardware. No obstante existe la posibilidad de destinar una máquina GNU/Linux para que actúe como balanceador por software.

A continuación se plantea una posible escenario de escalabilidad del sistema, dividido en varias máquinas:

- La primera máquina posee el conjunto de servidores que escuchan en el puerto 1 y cachea los resultados de la geolocalización de las peticiones en la cache 1 y guarda los mapas en el respositorio1. Además esta máquina corre un servidor web 1 para proporcionar todos los mapas solicitados a todos los dispositivos móviles que le realicen peticiones.
- Ante un aumento considerable de dispositivos móviles en el sistema, el servidor de comunicaciones es incapaz de atender de forma fluida a todas las peticiones, por lo que se decide poner en marcha otra máquina cuya diferencia radica en que posee dos instancias de servidores de comunicación, geolocalización y mapas escuchando en los puertos 2 y 3 junto con sus respectivas caches, repositorios de mapas y servidores web
- Debido a otro aumento de dispositivos móviles, dada la gran acogida del sistema turístico, se opta por incorporar una nueva máquina cuyos servidores de comunicación, geolocalización y mapas escuchan en el puerto N, pero las caches de geolocalizaciones y el repositorio de mapas son las utilizadas en el la segunda máquina. Además esta tercera máquina tiene su propio servidor web para atender las peticiones de las URLS de los mapas.

Este es un claro ejemplo de como puede escalase el sistema incorporando distintas máquinas cada una corriendo los servidores que se estimen oportunos y además compartiendo recursos.

No obstante es altamente recomendable compartir los recursos de cache de geolocalizaciones y repositorio de mapas, para que el sistema no tenga redundancia de datos en distintas ubicaciones.

Si existiesen diferentes caches de geolocalizaciones y repositorios de mapas, sería altamente recomendable incorporar procesos (objetivo de nuevas funcionalidades) que permitan reestructurar y consolidar esas caches y repositorios para que siempre tengan las misma información.

A continuación se muestra la figura correspondiente al planteamiento de escalabilidad anterior.

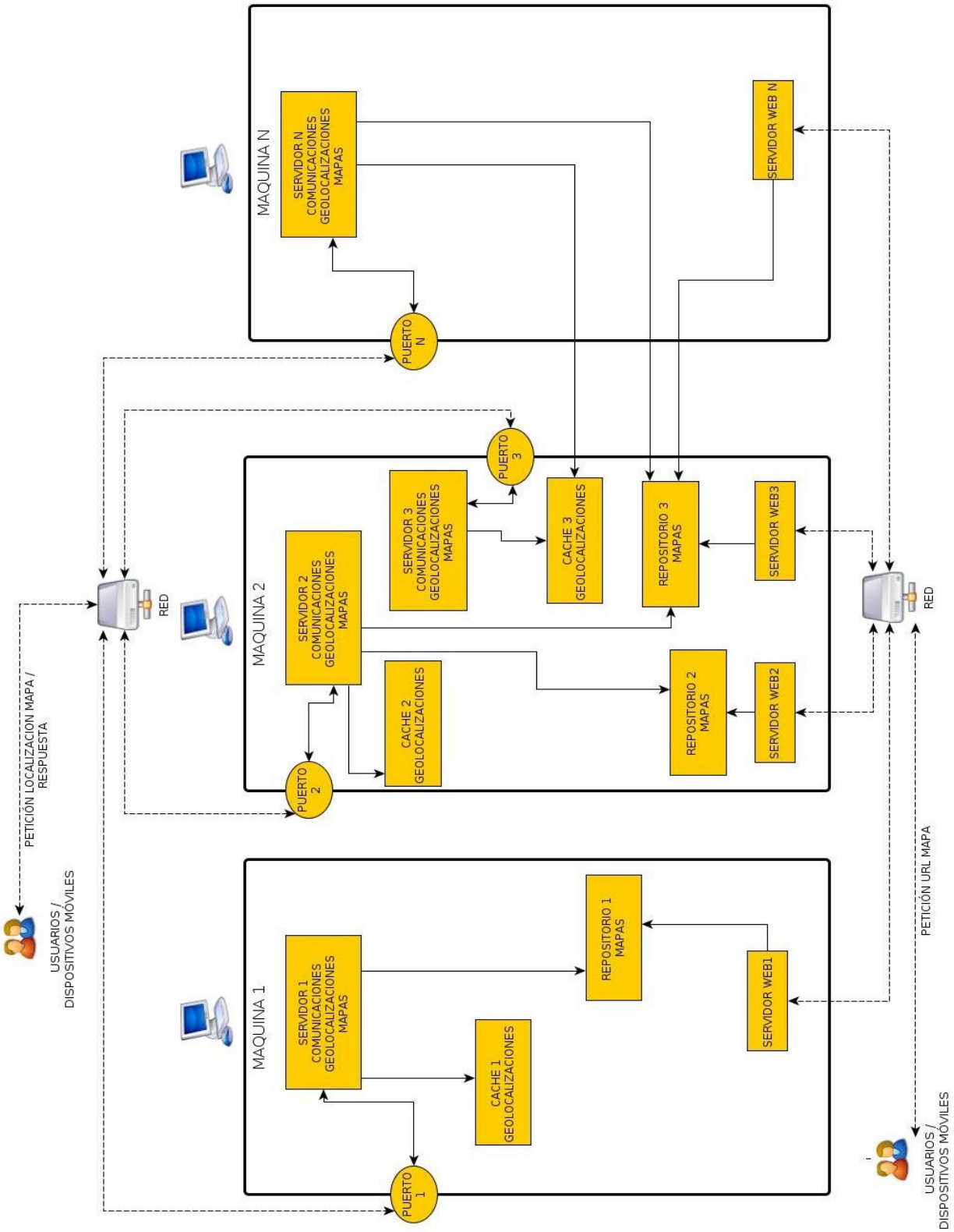

Figura 29: Escalabilidad del MGM

7.1 Anexo. Implementación MGM como servicio web

Tras la implementación de una primera versión del MGM, se llegó a la conclusión de poder implementar el MGM de tal forma que no fuese necesario el uso de sockets para interactuar con él y, los servidores de comunicación, geolocalización y mapas, no tuviesen que estar obligatoriamente a la vez, en una o varias máquinas como un bloque conjunto, dependientes los unos de los otros.

Es por ello por lo que una vez implantado el sistema se ha decidido además separar los tres servidores a modo de servicio web que no dependa del uso de sockets para su puesta en explotación de forma totalmente independiente en tantas máquinas como se quiera. De esta manera ante funcionalidades futuras o la utilización de este sistema en otros proyectos, se pueden realizar peticiones vía web tanto al geolocalizador como al módulo de mapas.

En la siguiente figura se muestra la implementación del sistema de forma totalmente independiente a modo de servicio web.

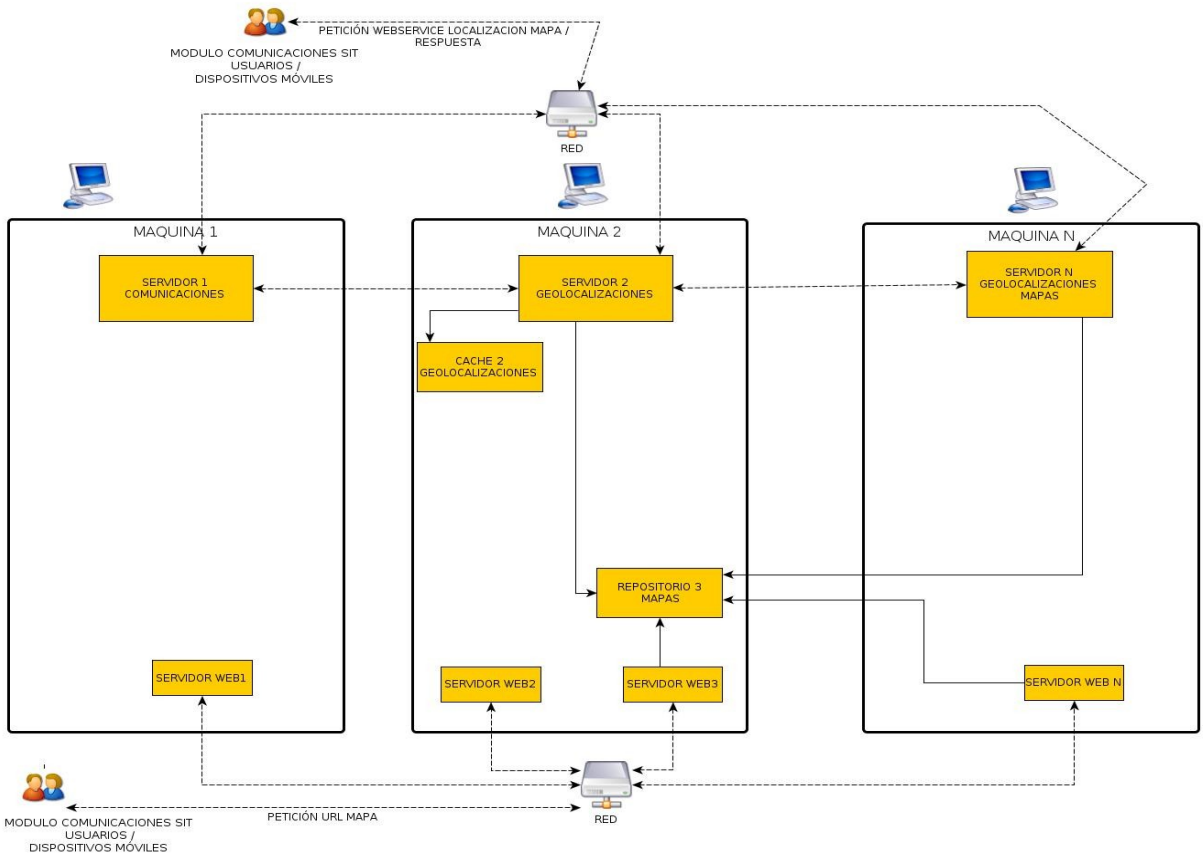

Figura 30: MGM como webservice

8. FUTURAS LÍNEAS DE TRABAJO

El diseño modular del MGM permite que en un futuro nuevas especificaciones o funcionalidades, se puedan implementar de manera totalmente independiente sin variar el código fuente del núcleo del MGM.

No obstante sería recomendable poder aplicar nuevas filosofías, tecnologías y funcionalidades tales como:

- Utilización de Memcached; además de las caches utilizadas hasta ahora todas ellas en disco, se trata de almacenar cache en memoria RAM, incrementando de forma espectacular el acceso y escritura de información en la misma. Para más información se puede visitar en http://memcached.org.
- Añadir Google Maps como servidor de geolocalizaciones y mapas remoto, complementando al actual.
- Posibilidad de poder ofrecer y visualizar rutas en los mapas, a modo de polígonos.
- Implementación de los resultados de los mapas en un navegador en los dispositivos móviles, permitiendo el desplazamiento y recalculo del mapa y ruta, ante la proliferación de dispositivos móviles con pantallas táctiles y el abaratamiento de los costes de conexión.
- Nuevos sistemas de almacenamiento como TokyoCabinet (http://fallabs.com/tokyocabinet/) para sustituir Berkeley DB.

9. MANUAL DE USUARIO

En este capítulo se detallan todos los pasos necesarios para la instalación del entorno y las librerías y el código fuente del MGM para su correcta implementación.

El punto de partida se establece en un sistema GNU Linux instalado en un PC, recomendando la distribución Ubuntu[30] utilizada para la implementación del presente PFC.

Para configurar el entorno y herramientas necesarias para la ejecución de MGM se instalarán una serie de paquetes contenidos en una serie de repositorios ubicados en servidores remotos configurados en la distribución GNU Linux instalada.

Para la descarga e instalación de los paquetes necesarios, se podrá utilizar una herramienta por línea de comando -ejecutada en una consola- que permite la descarga de dichos paquetes. Existen también no obstante, interfaces gráficas que permiten instalar los paquetes como el software "Synaptic" instalados por defecto en distribuciones Ubuntu.

Para la instalación de paquetes por línea de comando utilizando una terminal o consola de Linux utilizaremos el comando "apt-get install" seguido de los nombres de los paquetes a instalar. La ejecución de este comando se debe realizar como usuario "root", ya que se necesitan privilegios administrativos para la instalación de paquetes en Linux.

9.1 Instalación y configuración de Php versión 5.3

Autenticado en la consola como usuario root, se debe teclear "apt-get install php5 php5-cli php5-curl" para instalar por defecto php versión 5.3 en el sistema. Tras la descarga de los binarios, Linux procederá a su configuración por defecto. Normalmente los binarios de php se instalan en el directorio /usr/bin.

Para comprobar que se ha instalado correctamente se debe teclear en la consola "php -i" y se puede comprobar como php devuelve los valores configurados por defecto y las variables de entorno del sistema.

En el caso de que se deba configurar directivas y opciones de php distintas a las utilizadas por defecto, es necesario editar el archivo /etc/php5/cli/php.ini y configurar las opciones definidas en él.

El paquete php5-cli nos permite utilizar php por línea de comando, necesarias en las órdenes para poner en marcha el MGM.

El paquete php5-curl permite realizar conexiones remotas a URL de servicios remotos, tales como Yahoo Maps, para descargar información relativa a geolocalización y mapas.

El directorio de ficheros logs por defecto se encuentra en /var/log/php/. El más utilizado es error.log, que permite ver cada uno de los errores y avisos en la ejecución de cualquier script o programa de php por línea de comandos.

9.2 Instalación de librerías de Berkeley DB

Para el uso de estás librerías es necesario descargar el paquete libdb-4.6 o cualquiera superior. Para ello basta solo con teclear “apt-get install libdb-4.6” y automáticamente Ubuntu instalará los paquetes necesarios y configurará el uso de Berkeley DB en el sistema y, por consiguiente permitirá su uso en PHP.

9.3 Instalación y configuración del servidor web Cherokee

Para la instalación de este servidor web, basta solo con teclear “apt-get install cherokee”. Una vez instalado, se procederá a la configuración de ciertos datos tales como:

- puerto de escucha del servidor: por defecto el puerto 80. Es muy importante anotar este valor, ya que el necesario indicarlo en el MGM, concretamente en el fichero de configuración del servidor de comunicaciones, en la línea que detalla el valor de HOST_SERVIDOR_WEB_MAPAS, indicado la IP del servidor web seguida por dos puntos y el puerto de escucha del servidor web. Ejemplo 192.168.3.2:80, indica que que servidor web se ejecuta en la máquina con la dirección ip 192.168.3.2 en el puerto 80.
- document root: directorio base de ficheros que servirá el servidor web. Por defecto se establece a /var/www, pero se debe poner el path del directorio que contendrá los mapas descargados, configurado en el archivo de configuración del servidor de mapas en la línea indicada por la clave PATH_REPOSITORIO_DATOS_MAPAS. Ejemplo /mnt/filer-local/html/desarrollo/docs/MGM/mapas/yahoo_maps indica que ante cualquier petición al servidor de un mapas, si existe la imagen, se encontrará dentro del directorio anteriormente

indicado.

Para configurar estos dos valores, podemos editar el archivo de configuración de Cherokee, ubicado en la ruta /etc/cherokee/cherokee.conf.

Otra opción más cómoda, es la configuración de estos valores utilizando una interfaz gráfica que brinda Cherokee. Para ello se debe teclear "cherokee-admin" en un terminal o consola, obteniendo por pantalla una URL junto con usuario y una contraseña temporal para poder configurar estos valores y otros más desde un navegador. Tras introducir la URL en un navegador y una vez logueado con los datos anteriores, se puede proceder a la configuración de estos valores, tal y como se puede ver en las siguientes figuras:

Figura 31: Configuración de puerto de escucha

Figura 32: Configuración de servidor virtual

Es muy importante tener en cuenta que el usuario que ejecute por línea de comando del MGM, tenga permisos de lectura y escritura en el path que indica el document root del servidor web.

Si se desea cambiar el usuario y grupo por defecto que ejecuta el servidor web, se puede realizar desde la opción de datos generales en la pestaña de permisos de servidor, tal y como se indica en la siguiente figura:

Figura 33: Configuración de usuario y grupo que ejecutará el servidor web

En el caso de que se decidiese cambiar el usuario o el grupo, se deben cambiar el usuario propietario y grupo en los archivos de log del servidor web ubicados en /var/log/cherokee. Para ello basta con introducir el siguiente comando en una terminal "chown -R usuario:grupo /var/log/cherokee".

9.4 Instalación y configuración de las librerías y scripts del MGM

Para facilitar la instalación del MGM, dado que es necesario la creación de múltiples directorios para ubicar las librerías, los datos, las caches y los mapas, se ha optado por generar un fichero comprimido en formato tar.gz.

Simplemente basta con introducir en un terminal "tar -xvzf / archivo_MGM.tar.gz" y se instalará en el directorio por defecto, /mnt/filer-local/html/desarrollo/

Antes de introducir el comando anterior el usuario debe cerciorarse que está logueado como el usuario configurado con permisos de ejecución en el servidor web.

9.5 Puesta en marcha y ejecución del módulo MGM

Para la ejecución del MGM, simplemente basta con ejecutar en un terminal el comando" sh /mnt/filer-local/html/desarrollo/ejecutables/MGM/ejecuta_servidor.sh", y el servidor se pondrá en marcha.

Si en cualquier momento se desea matar el proceso, sabiendo su identificador de proceso "pid" en Linux mediante el comando "ps -ef | grep servidor" y a continuación como usuario root ejecutar el comando "kill numero_proceso".

10. REFERENCIAS

1 Geolocalización, georrefeenciación: posicionamiento con el que se define la localización de un objeto espacial (representado mediante punto, vector, área, volumen) en un sistema de coordenadas y datum determinado. Para más información puede visitar http://es.wikipedia.org/wiki/Georreferenciacion.

2 XML: eXtensible Markup Language (lenguaje de marcas extensible). Para más información visite http://es.wikipedia.org/wiki/Extensible_Markup_Language.

3 LAN: Locat Area Network o red de área local.

4 WAN: Wide Area Network o red de área amplia. Son redes informáticas que se extienden sobre un área geográfica extensa.

5 IBM: International Business Machines. Más información en http://www.ibm.com.

6 ITU-T: Unión Internacional de comunicaciones.

7 ISO: Organización internacional para la estandarización

8 W3C: World Wide Web Consortium. http://www.w3c.com

9 URI: Universal Request Identificator. Un URI es una cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema.

10 WSDL: Web Service Description Language. Es un formato XML que se utiliza para describir servicios Web

11 SOAP: Simple Object Access Protocol. Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

12 HTTP: Hiper-Text Transport Protocol. Es el protocolo usado en cada transacción de la World Wide Web.

13 GNU:El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre: el sistema GNU. Para más información puede visitar http://www.gnu.org/philosophy/free-sw.es.html

14 Microsoft: http://www.microsoft.com.

15 Apple: http://www.apple.com.

16 Java : http://www.java.com

17 Sun Miocrosystems: http;//www.oracle.com

18 PHP: http://www.php.net

19 Free Pascal: http://www.freepascal.org

20 Mysql: http;//www.mysql.com

21 PostgreSQL: http://www.postgresql.org/

22 Firebird: http://www.firebirdsql.org/

23: BerkeleyDB: http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html

24: Oracle: http://www.oracle.com

25 ACID: En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Así pues, si un sistema de gestión de bases de datos es ACID compliant quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID. En concreto ACID es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en castellano.

26 Apache: http://www.apache.org

27 Lighttpd: http://www.lighttpd.net/

28 Cherokee: http://www.cherokee-project.com/

29 URL: Localizador uniforme de recursos, más comúnmente denominado (sigla en inglés de Uniform Resource Locator), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

30 Ubuntu: Distribución GNU Linux muy usada en la actualidad desarrollada por la empresa Canonical. Más información en http://www.ubuntu.com

10.1 Fuentes consultadas

Para la información sobre servicios web se han consultado las siguientes fuentes

- Título: Restful web services Cookbook . Autor: Subbu Allamaraju. Editorial:O' Reilly.
- Título: Servicios Web XML. Autor: Cauldwell, Patrick. Editorial: Anaya Multimedia
- http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb
- http://www.desarrolloweb.com/articulos/1852.php
- http://petra.euitio.uniovi.es/~i1893878/pmwiki/pmwiki.php?n=ServiciosWeb.Introduccion

Información que versa sobre el software libre y sus aplicaciones

- http://www.gnu.org/
- http://www.proposicion.org.ar/doc/razones.html
- http://es.wikipedia.org/wiki/Software_libre
- Tiulo: Software libre para una sociedad libre . Autor:Richard M. Stallman

Información referente a los sockets y su programación

- http://devzone.zend.com/node/view/id/1086
- http://www.programacion.com/articulo/smtp_utilizando_sockets_en_php_176
- http://www.elguruprogramador.com.ar/articulos/introduccion-a-los-sockets-en-php.htm
- http://www.programacionweb.net/articulos/articulo/?num=159
- http://www.nociondigital.com/webmasters/php-tutorial-sockets-en-php-detalle-181.html
- http://php-es.com/ref.sockets.html
- http://www.chuidiang.com/clinux/sockets/sockets_simp.php
- http://www.oracle.com/technetwork/java/socket-140484.html

Especificaciones tecnologías y protocolo TCP/IP

- Título: Redes de computadores, tercera edición: Autor: Andrew Tannembaum

Editorial:Pearson Educación.

- Manual de Redes, Routers y Switches de CISCO Systems

Lenguiajes de programación como PHP, Delphi y Lazarus Java

- http://es.wikipedia.org/wiki/PHP
- http://www.php.net/
- http://es.wikipedia.org/wiki/Delphi
- http://es.wikipedia.org/wiki/Lazarus
- http://wiki.lazarus.freepascal.org/Overview_of_Free_Pascal_and_Lazarus/es
- http://es.wikipedia.org/wiki/Lenguaje_de_programacion_Java

Servidores web Cherokee, Lighttpd y Apache

- http://www.cherokee-project.com/
- http://www.cherokee-project.com/benchmarks.html
- http://www.cherokee-project.com/doc/basics_why_cherokee.html
- http://www.apache.org/
- http://linux.ciberaula.com/articulo/linux_apache_intro/
- http://www.lighttpd.net/

Base de datos Berleley DB

- http://www.bulma.net/body.phtml?nIdNoticia=1472
- http://download.oracle.com/docs/cd/E17076_02/html/toc.htm
- http://www.unixuser.org/~euske/doc/cdbinternals/index.html

Base de datos Postgresql

- http://www.postgresql.org/docs/
- http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html

Base de datos Mysql

- http://dev.mysql.com/
- http://dev.mysql.com/doc/